uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería Informática

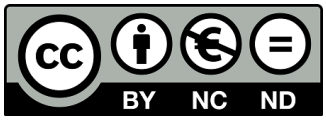
2021-2022

*Apuntes*

# Algoritmos Genéticos y Evolutivos

---

Jorge Rodríguez Fraile[1]





[1]Universidad: 100405951@alumnos.uc3m.es | Personal: jrf1616@gmail.com

# ÍNDICE GENERAL

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# 1. INFORMACIÓN

## 1.1. Profesores

Magistral: Pedro Isasi, pedro.isasi@uc3m.es, 2.1.B13.

Práctica: Yago Sáez, yago.saez@uc3m.es, 2.1.C13

Miércoles clases Práctica / Jueves clase Magistral.

Las tutorías solicitarlas por correo electrónico, si no responden en 24-48 horas volver a enviar el mensaje, mejor insistir que perder el tiempo ante posible perdida.

Se utilizará portátil en las clases prácticas, traerlo para ir resolviendo los ejercicios de programación.

## 1.2. Objetivos

- Trataremos de aprender métodos para replicar fundamentos biológicos, mediante programación evolutiva (Veremos estrategias históricas de la evolución, como Darwin o Mendel), para resolver problemas del campo de la Inteligencia Artificial.
- Se estudiarán y analizarán distintas técnicas de computación evolutiva que se basan en distintos paradigmas biológicos.
- Lo que buscamos es ser capaces de entender cómo funciona la inteligencia de las personas o animales para ver cómo se desarrollan los procesos y poder programarlos para que hagan una tarea o aprendan a hacerla.

Un ejemplo de aplicación consiste en desarrollo de antenas que se generan según distintas mediciones de señal de manera que se extenderá en aquellas direcciones que la maximicen, también evitando las interferencias. Este ejemplo se basa en las plantas, en la manera que tienen de ir buscando la luz para poder obtener energía y seguir creciendo.

También se emplean muchas basadas en insectos por su corta vida, que permiten observar la evolución de una manera más rápida.

Inteligencia artificial (no hay una única definición): Disciplina que consiste en diseñar programas que son capaces de resolver problemas de manera similar a un humano (razonamiento, aprendizaje o creatividad), pero sin haberles programado el proceso que les permite resolverlo, mediante una serie de casos base. De manera que dándole unos datos sea capaz de resolverlo, aprende a resolverlos.

### 1.3. Evaluación continua

- 3 prácticas a lo largo del curso, que podrían ser 2 prácticas y una extensión de la primera.
    - Consistirán en la entrega de una memoria siguiendo unos puntos dados y el código.
    - Individuales, excepto la última que es en equipo. 1 punto, 1.5 y 2.5 puntos
- 2 pruebas de evaluación, 2,5 puntos cada una.
- Examen final no obligatorio, si no se presenta a este la nota final será la de evaluación continua. Si te presentas se pondera la nota final tal que el 0.25 es evaluación continua y el 0.75 la nota del examen final.
    - Sin hacer el examen final:
        - 25 % Parcial 1
        - 25 % Parcial 2
        - 10 % Práctica 1
        - 15 % Práctica 2
        - 25 % Práctica 3
    - Haciendo el examen final
        - 6,3 % Parcial 1
        - 6,3 % Parcial 2
        - 2,5 % Práctica 1
        - 3,8 % Práctica 2
        - 6,3 % Práctica 3
        - 75 % Examen final

# 2. TEMA 1: INTRODUCCIÓN A LOS ALGORITMOS DE COMPUTACIÓN EVOLUTIVA

## 2.1. Teoría genética

Este tema nos permite entender de donde surge la intención de tratar de imitar los comportamientos biológicos y las teorías evolutivas que hay, en particular la evolución. Lo que trataremos es de acercarnos a como la biología es capaz de crear sistemas complejos.

### 2.1.1. Los cuatro pilares de la evolución

Dobzhansky (1973): "nothing in biology makes sense except in the light of evolution".

- **Población:** El conjunto de individuos.
- **Diversidad:** Es la variedad de individuos, que difieren en cualquier cosa, la naturaleza es la que se encarga de mantenerla y tiende a generar nueva diversidad. Las leyes de la naturaleza se dirigen a crearla.
- **Herencia:** Esencial para que se puedan transmitir la información entre individuos.
- **Selección:** Debe producirse la muerte de algunos individuos para que pueda seguir adelante la evolución, es fundamental.

### 2.1.2. Selección Natural

No es selección de los mejores, es un proceso estocástico, pueden sobrevivir individuos no tan buenos individualmente pero si para el colectivo, porque mejoren la reproducción media.

La evolución no tiene un objetivo, no es un progreso, solo se adapta a las circunstancias y al medio. No necesariamente una generación en mejor que la anterior.

Actúa en el aquí y ahora, lo que es bueno en términos reproductivos aquí o ahora, puede no selo allí o después.

La vida no tiende a la complejidad, hay muchos más individuos simples que complejos.

En el mejor de los casos, la combinación de variedad, herencia y selección puede incrementar "hoy" la proporción de individuos cuyos padres disponían de características más propicias "ayer".

### 2.1.3. De donde viene la diversidad

Viene de que la reproducción NO es perfecta, esto permite que haya ciertas imperfecciones (muy poco frecuentes) que les hagan más viables reproductivamente y se transmitan a sus descendientes, eso genera diversidad al haber pequeñas variaciones.

Aquellas características que no son ventajosas, pero tampoco afectan negativamente, son transmitidas también.

Es cuestión de estadística que se produzcan las variaciones, pero este proceso repetido innumerables veces, genera diversidad.

Permite al individuo "ensayar" nuevas funcionalidades, comportamientos, morfologías, nuevos nichos y propagar a generaciones futuras aquellos que mejor se adapten al entorno.

### 2.1.4. Escala Evolutiva

La vida se dio muy al comienzo de la vida de la Tierra, pero se han dado muchas grandes extinciones, una vez la vida se da la vida se va sabiendo abrir camino.

3.500 Ma para generar los primeros seres eucariotas pluricelulares.

Solo 500 Ma para generar el resto de los seres vivos (80 % contra 20 %).

Los organismos microscópicos, similares a aquello que evolucionaron tempranamente continúan siendo altamente exitosos y dominan la Tierra.

La mayor parte de las especias y la biomasa terrestre está constituida por procariotas.

### 2.1.5. Primeros pasos

**Moléculas replicantes → poblaciones de moléculas en compartimentos**. Favorece la cooperación entre replicantes, al estar dentro de una membrana común

**Replicadores independientes → cromosomas**. Los replicadores independientes se unen formando estructuras, cromosomas, que facilitan su supervivencia

**ARN como gen y enzima → ADN y proteína**. Se separa la información de los procesos enzimáticos

**Nace el código genético**. División del trabajo, aumento en la complejidad de lo producido

### 2.1.6. Estromatolitos (-3500 Ma)

Agrupaciones de células unicelulares en forma de colonias capaces de generar con energía solar un gas tóxico, $O_2$, de forma masiva.

A pesar de este gas tóxico la vida se adaptó para utilizar ese oxígeno.

### 2.1.7. Célula eucariota (-3500 Ma a -1200 Ma)

**Procariota → Eucariota**. Aparece un núcleo celular, donde está la información y orgánulos, esto ayuda a la supervivencia de la célula.

### 2.1.8. Organismos pluricelulares (-540)

Antes del cámbrico había organismos simples, paquetes de proteínas autoreplicantes tipo medusa o esponjas habitando el mar.

Se alimentaban de bacterias o filtrando el agua, no había plantas.

### 2.1.9. El Cámbrico

Aparecen cadenas tróficas complejas (animales que comen otros animales), todos los grandes filos (grandes divisiones de la naturaleza) de la naturaleza (50 en el cámbrico, antes solo había 1 y después 8 más, tras esto solo quedaban 20).

Cambió la faz de la tierra y conformaron la vida tal y como lo conocemos hoy.

Ocurrió en solo 5 millones de años.

### 2.1.10. Periodo Cámbrico

Antes del cámbrico organismos simples, paquetes de proteínas autoreplicantes tipo medusas o esponjas habitando el mar.

Se alimentaban de bacterias o filtrando el agua. No había plantas.

Aparecen cadenas tróficas complejas (animales comen otros animales).

De repente aparecen todos los grandes filos de la naturaleza, 50 en el cámbrico, solo 1 antes y 8 después (solo quedan 20).

Mares casi inertes en mares repletos de vida. Cambió la faz de la tierra y conformaron la vida tal y como la conocemos hoy.

Ocurrió en solo 5 millones de años.

Causas ambientales. La escasez de recursos dio lugar a una competencia ecológica que favoreció la coevolución “depredador- presa”.

Causas geológicas. La separación de los continentes permite la segregación de las poblaciones y la diversificación evolutiva.

Causas morfológicas. Aparición del mesodermo.

Permite la generación de estructuras de colágeno (piel, cartílago, músculos).

Esqueletos rígidos (huesos, conchas).

Proliferación de oxígeno. Los animales no están obligados a obtener el oxígeno por todo el cuerpo, sino por órganos específicos, dejando el resto para caparazones o exoesqueletos.

### 2.1.11. Seres complejos

**Clones asexuales → Poblaciones sexuales**, es el mecanismo que guía el proceso, permite generar más individuos y diversidad de una manera eficaz. Aparece el sexo.

**Protista → Animales, plantas y hongos**. Aparece la diferenciación celular, todas las células tienen todo el material genético. Cada tipo de célula tiene activa una pequeña parte de su material genético. Empiezan a aparecer organismos.

**Individuos solitarios → Colonias**, conjunto de organismos que funcionan como uno. Organismos sociales.

### 2.1.12. Últimos pasos

-500 Ma plantas y hongos colonizan la tierra.

Poco después aparecieron artrópodos y otros animales

-300 Ma aparecen los anfibios.

Seguidos por los primeros amniotas (reproducción ovípara en medio seco terrestre)

-200 Ma Mamíferos.

-100 Ma Aves.

-2 Ma Hombre. Algo peculiar de los humanos y simios es que son consciente de su propia existencia a diferencia de otros animales.

### 2.1.13. Las leyes de Mendel

Hizo experimentos con guisantes entre 1857 y 1868. Publico su primer artículo en 1866.

Se fijó en dos características, la altura de la planta y la rugosidad del guisante.

Mendel cruzó artificialmente plantas altas con bajas, y siempre se producían plantas altas.

Después cruzó las plantas resultantes y obtuvo unas proporciones de 3 a 1.

Elaboró una teoría corpuscularia de la herencia, que decía que la herencia está en alguna parte del individuo codificada.

Envió los resultados a un científico famoso, Nageli, pero pasó desapercibido.

50 años después se redescubrió su teoría y fue aceptada.

### 2.1.14. El código genético

Watson y Crick descubrieron en 1951 la estructura en doble hélice del ADN: Adenia-Timina Guanina-Citosina.

Más tarde se descubrió el ARN. Igual excepto Uracilo en vez de timina.

**Cada tres bases (codón) → un aminoácido**

El ARN no tiene estructura fija en doble hélice. Puede doblarse de muchas formas.

Los pliegues le dan carácter de enzima, y es además portador de código genético.

Hay $4^3 = 64$ posibles codones para 20 aminoácidos. Tres de ellos no codifican aminoácidos, sino controles (STOP).

Más de un codón codifica un mismo aminoácido. Los aminoácidos más numerosos son codificados por más codones, y parecidos. Esto evita las mutaciones destructivas.

Los aminoácidos forman las proteínas.

Las proteínas son enzimas que gobiernan todos los procesos químicos.

### 2.1.15. La síntesis de proteínas

El ARN no tiene estructura fija en doble hélice. Puede doblarse de muchas formas.

Los pliegues le dan carácter de enzima, y es además portador de código genético.

El ADN contiene el material genético, es copiado “en negativo” por el ARN-mensajero.

El ARN-mensajero pasa la información al ARN-de transferencia.

El ARN-de transferencia tiene una zona desplegada compuesta por tres bases y tiene adosado un aminoácido.

El ARN-t se “pega” al trozo de ARN-m complementario, cuando el anterior trozo ya ha sido leído, y deposita su aminoácido a continuación del anterior.

Este proceso se realiza en presencia de una enzima llamada ribozima.

### 2.1.16. Definiciones

**Gen:** La definición no es clara. Todo aquello que produzca de forma monolítica una característica en un individuo.

**Genotipo:** Es el conjunto de todos los genes. Es la secuencia de bases enumerada una detrás de otra

**Fenotipo:** Es la consecuencia física, psíquica, de comportamiento, o cualquier otra; de un determinado gen

### 2.1.17. Reproducción

Existen dos tipos de reproducción: asexual y sexual

La asexual es cuando un individuo por sí solo produce copias idénticas de él mismo.

- P. ej. Las estrellas de mar

La sexual es cuando se necesitan dos individuos para producir un descendiente (o varios), y el resultado es un individuo cuyos genes son una mezcla de los de sus progenitores.

La reproducción sexual produce constantemente individuos diferentes y ayuda a la variedad genética

Se desconocen las causas de la aparición de la reproducción sexual Computacionalmente la reproducción sexual es mucho más potente

#### Haploidad

Tienen un solo juego de cromosomas.

Los individuos se reproducen por sí mismos.

La reproducción da lugar a individuos genéticamente idénticos al progenitor

#### Diploidad

Los cromosomas están pareados. Las células sexuales se generan mediante meiosis y son "haploides", conteniendo información redundante e incluso contradictoria.

Los descendientes reciben un cromosoma de cada uno de los progenitores para formar un embrión diploide

#### Haplo-Diploidad

Unos individuos son haploides (machos) y otros diploides (hembras).

- P. ej. Los insectos sociales

## 2.2. Conceptos de Computación Evolutiva

La evolución se apoya en 4 pilares: Población, Diversidad, Herencia y Selección.

### 2.2.1. Definición de CE

#### Qué es la Computación Evolutiva?

Se parte de una **población que es un conjunto de soluciones** aparentes al problema que buscamos resolver. Se emplea en aquellos problemas que encontrar una solución no es difícil, ya que hay una cantidad enorme de alternativas, pero si una que sea buena y eficiente.

Buscamos **encontrar una solución que sea suficientemente buena** (encontrar la óptima es muy difícil y costoso), este concepto debe ser definido para saber cuándo se alcanza.

El problema tiene que poder describirse como un conjunto de estados/soluciones alternativas, **encontrar una buena representación** de las mismas.

Se debe poder **medir la calidad de cada solución** alternativa al problema, mediante una **función de fitness**, para saber si es válida.

Desde el punto de vista de optimización hay que generar soluciones cada vez mejores.

Desde el punto de vista de búsqueda se trata de buscar buenas soluciones en un espacio de búsqueda inmenso (como si fuera infinito).

La búsqueda se realiza de forma poblacional (un conjunto de individuos), no mediante trayectoria de un solo individuo.

**Son metaheurísticas**. La heurística es siempre la misma (universal) por eso se pueden aplicar a una variedad de problemas y construir heurísticas específicas.

La heurística es una **versión reducida de la selección natural**, combinación de los buenos candidatos, se generan individuos a partir de los previos y nos vamos quedando con los mejores.

Son **técnicas estocásticas**, para 2 ejecuciones se pueden observar distintas salidas, ya que por probabilidad no se han realizado las mismas operaciones. Esto permite evitar el estancamiento.

### Evolución artificial

Se basa en los mismos principios que la evolución natural

- Mantenimiento de una **población**
- Creación de **diversidad**
- Un mecanismo de **selección**
- Un proceso de **herencia** genética

Se diferencia en que la evolución artificial es **orientada hacia una meta, no como la natural**.

Hay que obtener una **medida cuantitativa de la calidad del individuo (solución) para poder seleccionar al mejor y generar más copias** de los mejores individuos.

Es un proceso iterativo que finaliza cuando se ha obtenido un individuo lo suficientemente bueno.

### Procedimiento

Se parte de una población inicial de soluciones aleatorias.

1. Selección de progenitores, mediante la función de fitness evaluamos los individuos y aplicando reglas de selección escogemos una población de progenitores.
2. Reproducción, las operaciones son estocásticas y aleatorias.
3. Nueva población que se pasa por la función de fitness para ver si es mejor que la progenitora.
4. Evaluación de la descendencia, que vuelve al paso 1.

### Representaciones genéticas

**Genotipo:** La codificación del individuo.

**Fenotipo:** El individuo en sí, su significado.

Los individuos deben poder representarse de forma que:

- El individuo pueda ser decodificado fácilmente (fenotipo).
- Se puedan generar nuevos individuos mediante combinaciones de los que ya existen.
- El conjunto de todos los posibles individuos debe coincidir lo máximo posible con el de todas las soluciones posibles, la codificación se debe ajustar a todo el espacio de soluciones (no solo a una pequeña parte)
- Se debe evitar que se codifiquen soluciones no válidas, atender a las limitaciones.

Encontrar una buena representación es el aspecto clave del algoritmo.

**Pueden ser:** Discretas (Algoritmos Genéticos), continuas(EE, Evolución Diferencial) o basadas en árboles(Programación genética)

### 2.2.2. Población Inicial

Se trata de generar una población de n individuos (soluciones) de dimensión k de forma aleatoria.

Si se tiene algún conocimiento previo para generar una buena población inicial usarlo, pero si no se emplea:

- **Aleatoria uniforme:** Hay que tener en cuenta que habrá huecos en el espacio de soluciones no cubiertos, ya que es aleatorio. Genera valores aleatorios entre los rangos marcados para los valores del individuo.
- **Muestreo uniforme hipercúbico:** Para evitar que haya regiones más pobladas que otras se divide el espacio en n regiones iguales y se generan dentro de estas individuos aleatoriamente. De esta manera están más distribuidos uniformemente en el espacio.
- **Secuenciado simple inhibido:** Otra manera de buscar que este todo poblado es generar individuos que se encuentren al menos a una distancia mínima $\Delta$, se parte de un individuo aleatorio y se van generando más mientras están al menos a esa distancia del resto de individuos.
- **Heurística:** Emplea una heurística como podría ser la greedy.

### 2.2.3. Evaluación

Los individuos son evaluados por su capacidad para resolver el problema

**Evaluación cuantitativa:** Se asigna un valor numérico.

**Evaluación cualitativa:** No se produce un valor numérico, sino por comparativa con el resto.

La función de evaluación es lo que más tiempo de cálculo consume, hay que simplificarla al máximo.

Se tiene que adaptar al problema la función de fitness, no a un caso específico, dado que el algoritmo producirá soluciones tratando de maximizar la evaluación de los individuos, y ningún otro factor.

Hay que estar seguro de que lo que queremos maximizar u obtener, está descrito con precisión en la función de evaluación.

#### Paisaje de calidad

El poder tener una **representación gráfica de la calidad de los individuos** permite estimar la complejidad del problema

Muestra el valor de calidad de todos los puntos del espacio de estados, aunque dado el gran tamaño del espacio seran una **estimación**.

Se puede calcular la calidad de los individuos que rodean a uno determinado, e intentar extrapolar en el resto del espacio.

### 2.2.4. Selección

La selección tiene como objetivo que los mejores individuos generen más descendentes que los peores.

Se utiliza el término **presión selectiva** (cuando un individuo tiene mucho estrés encima para evolucionar, tiene competencia) como la cantidad de individuos diferentes que generarán descendencia.

Mucha presión selectiva indica que solo unos pocos generarán descendencia. Cuando no hay presión selectiva no necesita evolucionar sabe que podrá seguir igual.

**Tres tipos:**

- **Selección proporcional o Ruleta:** La probabilidad de que un individuo sea seleccionado para reproducción es proporcional al cociente entre su valor de evaluación, y el total de la población.

$$p(i) = \frac{f(i)}{\sum_{i=1}^{N} f(i)}$$

  Este método no funciona bien en dos situaciones:

  - Cuando un individuo es muy superior al resto (Problema del superindividuo), ya que estará sobrerepresentado y pierde diversidad.
  - Todos los individuos son igualmente válidos, todos tendrán la misma probabilidad y se aleatoriza la evolución. Por probabilidad unos individuos podrían ser seleccionados preferentemente.

- **Selección por rango o jerárquica:** Probabilidad en función de la posición que ocupa en orden de mayor evaluación. Evita los inconvenientes de la ruleta o selección proporcional. La probabilidad la da la posición no el valor de la evaluación.

  Aunque un individuo sea muy superior no será una diferencia excesiva de probabilidad de selección, lo mismo si son parecidos. La presión selectiva vendrá dada por la diferencia entre posiciones y puede ser regulada.

- **Selección por rango o jerárquica truncada:** Se seleccionan una proporción de los mejores individuos, asignando la misma probabilidad de selección, y al resto cero.

  Los mejores se reproducen aproximadamente el mismo número de veces, el resto no transmite su material genético.

- **Selección por torneos:** Consiste en realizar un torneo entre un número reducido de individuos de la población, cada vez que haya que seleccionar. Se eligen T individuos de forma aleatoria uniforme y se selecciona el mejor, hay remplazo, es decir uno elegido puede volver a ser elegido.

  Se repite el proceso tantas veces como individuos a seleccionar.

  Cuanto más grande sea T más presión selectiva, se empieza con poca presión y se va aumentando más adelante.

  No es necesaria función de evaluación, solo comparar individuos.

  Es fácil pasar de exploración a explotación.

  Permite mantener y regular la variabilidad genética.

  Se elimina el efecto de los superindividuos.

  Se ajusta fácilmente a problemas multiobjetivo.

### 2.2.5. Emparejamiento

La reproducción es sexual, se realiza entre dos individuos, y la descendencia puede ser de un único individuo o de dos.

Los individuos se eligen de entre los seleccionados en el paso anterior según las siguientes maneras:

- Se eligen ambos de manera aleatoria.
- El primero aleatoriamente, el segundo por afinidad (genotípica o fenotípica).
- Se selecciona el más afín, o el mejor de los que están a proximidad (vecindario), o aleatoriamente proporcional a la calidad de los que están a proximidad

Una vez reproducidos, los individuos no vuelven a ser utilizados de nuevo para reproducción.

### 2.2.6. Reproducción

**Recombinación**

- **Discreta:** Se generan descendientes seleccionando una secuencia consecutiva de un progenitor, y la otra del otro.

  100 101 y 010 110 dan 100110 y 010101

- **Uniforme:** Se generan descendientes seleccionando los genes homólogos de sus progenitores aleatoriamente. Se escoge aleatoriamente los genes teniendo en cuenta su posición.

  10 01 01 y 01 01 10 dan 100110 y 010101.

- **Aritmética:** Se hace la media de los valores de los genes de los progenitores para generar un nuevo individuo. 0,2 0,6 1,2 y 0,4 0,4 1,0 dan 0,3 0,5 1,1

- **Secuencias:** El nuevo individuo mantiene la secuencia de los padres sin repetir valores. Se cogen secuencias de valores sin tener en cuenta la posición y van rellenando el progenitor.

  ABCDEFG y GFCDBAE dan E BCD GFA (la AE del segundo)

- **Intermedia:** La recombinación aritmética genera siempre valores entre los de sus progenitores, esto reduce la variabilidad y hace converger hacia valores intermedios definidos aleatoriamente en la población inicial.

  Esto se puede evitar permitiendo generar valores en un rango mayor al definido por los genes de los progenitores

No se realiza la media aritmética, sino un valor aleatorio en un rango extendido.

A partir de los valores de los genes de los progenitores, se obtiene un valor intermedio para cada descendiente

Para cada gen (i)

1. Se genera un valor aleatorio ($a_i$) en el intervalo [-d, 1+d], siendo d un parámetro del método.
2. Se asigna como valor del gen descendiente ($z_i$) la combinación de los valores de los genes homólogos de los progenitores ($x_i$, $y_i$). Se multiplica el gen del primer progenitor por $a_i$ y el del segundo por (1 - $a_i$) y se suman.

Si d=0 los genes de los descendientes generarán valores intermedios de los de los progenitores, que sería como recombinación discreta. Para evitar que los valores converjan hacia dentro, se asignan valores de d>0 así sale de las fronteras de sus progenitores.

- **Lineal:** Igual que la recombinación intermedia, pero con el mismo valor de a para todos los genes. Permite generar valores de descendientes en la línea definida por sus progenitores.

### 2.2.7. Inserción y remplazo

Una vez seleccionados los individuos a reproducir, emparejados y generados sus descendientes, hay que decidir cómo se van a renovar las poblaciones.

- Suponemos que el tamaño de la población debe permanecer constante de generación en generación, pero puede no ser así.
- **Política de inserción:** Qué individuos de los nuevos generados van a incluirse en la nueva generación.
- **Política de reemplazo:** Qué individuos de la población van a ser sustituidos por los nuevos.

#### Estrategias de inserción

Estrategia generacional:

- **Generacional:** Se renueva toda la población al completo.
- **Generacional elitista:** Mantiene los n mejores individuos de una generación a la siguiente ($n \ll \#population$) y renueva los restantes.

- **Elitista puro:** El mejor individuo de la población pasa a la siguiente, el resto se renueva.

Estrategia de estado estacionario:

- **Estado estacionario:** Se generan menos individuos del tamaño de la población y todos ellos se incluyen en la generación siguiente.
- **Estado estacionario puro:** Solo se genera un individuo nuevo en cada generación y es insertado en la siguiente generación.

Para estas hay que decidir qué individuos de la población serán sustituidos por los nuevos generados, con Política de reemplazo.

**Inserción híbrida:** Se generan más individuos nuevos que los que tiene la población.

- **Basada en fitness parcial:** Se seleccionan los n mejores individuos y se renueva la población por completo.
- **Basada en fitness:** Se seleccionan los mejores entre la población generada y la previa, de esta manera se renueva parcial o totalmente, depende del fitness.

**Estrategias de remplazo**

- **Globales:** Considerando toda la población.
  - Aleatoriamente
  - Los más antiguos de la población
  - Los peores de la población
  - Los más parecidos a los individuos a insertar
- **Locales:** De entre los que participan en la reproducción.
  - Los nuevos reemplazan a sus progenitores. n individuos como progenitores sustituidos por los n individuos nuevos generados.
  - El peor. n progenitores generan m individuos nuevos (m<n), de manera que los nuevos sustituyen a los peores de los n progenitores.
  - El más parecido. n progenitores generan m individuos nuevos (m<n), de manera que los nuevos sustituyen a los más parecidos de los n progenitores

### 2.2.8. Convergencia

Criterios de parada

- **Estática:** Se establece antes de empezar.
  - Número de iteraciones (generaciones).
  - Número de evaluaciones.
  - Un determinado tiempo.
- **Dinámica:** En cada momento se evalúa si merece la pena detenerse o continuar.
  - Cuando ya no se generen mejores soluciones.
  - La diversidad genética caiga por debajo de cierto umbral.

    Se calcula la semejanza genotípica entre todos los posibles pares de individuos de la población $D(P) = \sum_{i,j \in P} d(P_i, P_j)$.

    En función de la representación puede ser distancia Hamming (discreta), distancia Euclídea (continúa) o diversidad de entropía $D(P) = \sum_{k=1}^{l} \sum_{\alpha \in A} f_k(\alpha) \log f_k(\alpha)$ (continúa no binaria).
  - La diversidad fenotípica caiga por debajo de un umbral.
  - Mejor individuo, se para sí no se encuentra después de n iteraciones un individuo mejor.
  - Evaluación media (nos permite ver si hay posibilidad de mejora), calcula la media de la evaluación de los individuos de la población y cuando no mejora o muy poco (debajo de un umbral) se para.

# 3. TEMA 2: CONCEPTOS GENERALES DE ALGORITMOS EVOLUTIVOS

## 3.1. Elementos

**Codificación:**

- Individuo, cadena binaria que representa una solución al problema.
- Población, conjunto finito de soluciones.

**Operadores genéticos:** Reproducción, Cruce, Inversión y Mutación.

**Función de fitness:** Convierte un individuo en un valor real que evalúa la solución.

## 3.2. Diseño de un AG

Lo primero es plantearse si se puede resolver mediante esta técnica o no, si se puede entonces:

1. **Modelar el problema:** Elegir una buena representación es muy importante y diseñar una función de evaluación.
2. **Elegir los parámetros del método:** Que tipo de operaciones de selección y cruce, y asignar las tasas de probabilidad a los operadores.

## 3.3. Codificación

**Debe cumplir:** Diseño eficaz de genes y Alfabeto mínimo.

**Gen:** Secuencia genómica que produce una característica fenotípica. Deben ser pequeño y no estar relacionados entre ellos.

**Codificación binaria:** Cadenas de caracteres binarios. Se pueden representar números enteros y reales, si se determina la precisión (x bits parte entra y z parte decimal).

**Codificación binaria Gray:** Cadenas de caracteres binarias en la que la distancia Hamming entre codificaciones adyacentes es 1. Es conveniente para que fenotipos adyacentes se correspondan con codificaciones adyacentes.

**Codificación Discreta no binaria:** Cuando para cada gen hay un número de alternativas no muy grande, pero mayor que dos. Esta alternativa evita dejar valores en binario sin significado.

**Codificación de reglas:** Si condición, entonces acción. Se codifican por un lado las condiciones y por otro las acciones, sin olvidar los operadores ($\leq, \geq, \neq, =$). Las condiciones pueden ser números o atributos (se pone el valor del atributo, 1 o 0) y las acciones un conjunto predeterminado de valores.

- simetría bilateral, cuerpo segmentado, exoesqueleto y patas articuladas → es un [artrópodo]
- Si Si Si Si 9
- 1 1 1 1 1001

### 3.4. Operadores Genéticos

**Selección**, simula la selección natural darwiniana hasta completar una población:

- **Ruleta** (por valoración): Selecciona los individuos más aptos según la evaluación. Individuos con mayor evaluación generan mayor número de descendientes. La probabilidad de ser escogido es proporcional al fitness.

  Problemas:

  - Convergencia prematura: Aquellos individuos con una evaluación muy superior al resto tendrá muchos clones y la descendencia se parecerá mucho al progenitor, lo que lleva a que se estanque en una solución subóptima.
  - Estancamiento: Después de unas cuantas generaciones, todos los individuos tienen un valor de adecuación muy parecido entre ellos, por lo que hay baja presión selectiva al tener todos la misma probabilidad.

- **Jerárquica** (por posición): Se ordena la posición de mayor a menor fitness. Se selecciona al individuo proporcionalmente al rango que ocupa en la población. Maneras de calcular la probabilidad $P_i = \frac{p_i}{\sum_{j=1} p_j}$:

  - Esquema lineal, $p_i = A \cdot i + B$
  - Esquema exponencial, $p_i = A \cdot e^{B \cdot i + C}$

  Esto permite eliminar la influencia de individuos muy superiores al uniformizar las diferencias entre probabilidades por posición.

- **Torneos** (el mejor de un conjunto aleatorio de individuos).

  El tamaño del torneo permite variar la presión selectiva, además mantiene y regula la variedad genética.

  No necesita una función de evaluación, solo comparar individuos.

  Eliminar el efecto de los superindividuos.

**Cruce/Reproducción**: Simula la reproducción sexual.

- Se genera una población intermedia del mismo tamaño que la original, en la que tendrá más probabilidad de estar representados los mejores individuos.
- Puede ser: Simple (parte en 2), De dos puntos y (cada gen tiene una probabilidad de cambiar por el del otro progenitor).
- El problema del incesto, si solo se hace cruce puede ocurrir que se pierdan alelos (todos los individuos tengan el mimo valor para un gen). En este caso se pierde rápidamente la variedad genética y se produce convergencia prematura. Por esto se introduce mutación.

**Inversión**, operador a medida, adecuado para ciertos dominios.

**Mutación**, simula las mutaciones que se producen en la reproducción.

- Elige aleatoriamente un gen y cambia su valor, por lo que pueden producirse muchas o ninguna mutación en un individuo.
- Se emplean frecuencias bajas, para ir explorando poco a poco, pero si es demasiado baja hay incesto.
- Evita la perdida de alelos y aleatoriza la búsqueda.
- Se comienza con valores mayores (exploración) y se va reduciendo para afinar las soluciones (explotación).

### 3.5. Estrategias de reemplazo

Dependiendo del tipo de reemplazamiento de los descendientes en la población, entre generaciones consecutivas, se distinguen dos tipos de Algoritmos Genéticos:

- **Generacional:** Se reemplaza toda la población.
- **Estado Estacionario:** El reemplazamiento es parcial (n individuos). En este caso hay diferentes criterios de reemplazo:
  - Los n individuos con peor valor de adecuación son sustituidos por los nuevos.
  - Los nuevos reemplazan a sus n progenitores.
  - Los individuos más parecidos. Cada individuo nuevo reemplaza a aquel cuya distancia Hamming sea menor.
- **Estado estacionario puro:** Se reemplaza un único individuo.

## 3.6. Propiedades

### 3.6.1. Tamaño del espacio de búsqueda

Los algoritmos genéticos son útiles para tamaños grandes, cuando no hay manera de obtener la solución óptima.

Los tiempos en explorar las soluciones aumenta exponencialmente.

Son útiles a partir de un tamaño de entre 20 y 30, $2^{20}$ y $2^{30}$, los tamaños inferiores dan soluciones triviales y se pueden manejar.

Suponiendo que cada individuo tarda un milisegundo en evaluarse:

- Tamaño 20; $2^{20}$= 17,45 minutos.
- Tamaño 30; $2^{30}$= 12,42 horas.
- Tamaño 50; $2^{50}$= 35702 años.

### 3.6.2. Codificación

Es importante que trate de cumplir:

- **Completitud:** Todas las soluciones deben poder ser codificadas.
- **Coherencia:** Únicamente representar soluciones factibles, para no perder el tiempo.
- **Uniformidad:** Todas las soluciones con la misma cantidad de codificaciones, no hay privilegiados con más de 1 código asociado.
- **Sencillez:** Fácil y rápida de aplicar la codificación.
- **Adyacencia:** Pequeños cambios en los individuos representan pequeños cambios en la solución.

### 3.6.3. Limitaciones

- **Representación:** El espacio de búsqueda solo se representa mediante cadenas binarias o discretas, puede que esta no sea la más conveniente para el problema.
- **Dependencias:** No considera las restricciones o dependencias del problema.
- **Opacidad:** Solo está guiado por la aptitud de los individuos, no incorpora ningún otro conocimiento.
- **Finitud:** Solo puede trabajar con poblaciones finitas no muy grandes.

### 3.6.4. Problemas

- Pérdida de **variabilidad genética**.
- **Convergencia prematura**, estancamiento en mínimos locales.
- Problema de **epístasis**.
- Paisaje de la función de evaluación muy abrupto (**fitness landscape**).
- Problema de **desorientación** (deception).
- Problema de **deriva genética**.

### 3.6.5. Variabilidad genética

Es importante para evitar la convergencia prematura.

La generación de nuevos individuos se basa en la recombinación de los ya existentes.

La pérdida de variabilidad genética impide producir soluciones novedosas. Se pierde la capacidad de intercambio de información útil entre los individuos.

La similitud genética de los individuos indica el final del proceso de búsqueda.

Si la pérdida de variabilidad genética se produce en las primeras fases de la evolución se dice que hay convergencia prematura, que hace que el proceso se estanque en mínimos locales.

### 3.6.6. Convergencia prematura

En las fases tempranas de la evolución de un AG puede ocurrir que un individuo o un grupo de ellos obtengan una aptitud notablemente superior a los demás.

Riesgo de que se produzca una evolución en avalancha:

- Al incrementar los individuos más aptos, la diversidad disminuye, la siguiente generación se favorece más de estos hasta que dominan toda la población (superindividuos).
- Los superindividuos llevan hacia un subóptimo, son los más aptos en cierto momento.
- Puede ser deseable, en la fase final del proceso.

### 3.6.7. Epístasis

Cuando la búsqueda se vuelve más aleatoria, por dependencias entre genes.

La heurística de los AGs se fundamenta en la adyacencia e independencia de la codificación:

- Individuos genotípicamente parecidos producen fenotipos similares y viceversa.
- No existen interacciones entre los genes, son independientes.

Si la propiedad de adyacencia no se cumple, el proceso de búsqueda se hace menos heurístico.

Los nuevos individuos se generarán más aleatoriamente cuanto menos se cumpla la propiedad de adyacencia.

Si los genes dependen unos de otros, cuando se cambia el valor de uno, esto repercute en la característica fenotípica de otro. La búsqueda se hace también más aleatoria. Cuando esto ocurre se dice que la codificación tiene epístasis.

Esto es debido a la teoría de los bloques constructivos.

### 3.6.8. Geografía de la función de evaluación

Para que el AGs pueda encontrar una buena solución, los individuos deben tener evaluaciones suficientemente diferentes. En caso contrario se dice que el paisaje de la función de evaluación es demasiado abrupto.

Lo ideal es que haya un punto o zona en la que se produzca una subida para ir escalando. Que sean más suave los puntos con buena evaluación.

Puede haber problema que sean muy abruptos y no se puedan hacer con AG.

#### Función de evaluación abrupta

La forma de la función de evaluación es muy abrupta, lo que hace que no haya suficiente presión selectiva para aproximarse sucesivamente al óptimo.

Es difícil saber si se va por buen camino, todos los valores son muy similares y tienen las mismas probabilidades de sobrevivir, excepto unos pocos muy cercanos a la óptima.

La búsqueda se convierte en un proceso aleatorio

### 3.6.9. Desorientación

Al combinar dos buenos individuos se producen habitualmente individuos peores que ambos.

La teoría de los bloques constructivos necesita que se generen esquemas privilegiados.

Si bloques constructivos que generan por separado individuos buenos, al combinarse generan individuos peores, se dice que existe desorientación.

### 3.6.10. Deriva genética

Es un efecto estocástico que emerge del rol del muestreo aleatorio en la reproducción.

Se trata de un cambio aleatorio en la frecuencia de alelos de una generación a otra.

Existe una tendencia que te guía hacia soluciones no muy buenas.

Normalmente se da una pérdida de los alelos menos frecuentes y una fijación de los más frecuentes, resultando una disminución en la diversidad genética de la población.

Los efectos de la deriva se acentúan en poblaciones de tamaño pequeño, y ocasionan cambios que no son necesariamente adaptativos. Para evitarlo emplear poblaciones mayores.

### 3.6.11. Control de la diversidad genética

Para controlar la diversidad genética hay que actuar sobre el mecanismo de generación de descendientes.

Cuanto más se favorezca a los más aptos menor variedad se obtendrá, y aumenta el riesgo de perder la información de valor desarrollada por los individuos moderadamente aptos. Si no se favorece especialmente a los individuos más aptos, se obtendrá más diversidad a costa de hacer las etapas de selección menos eficaces.

A la mayor o menor tendencia a favorecer a los individuos más aptos se le llama presión selectiva, que es cuando un individuo tiene que luchar para pasar de generación.

La presión selectiva (PS) cuantifica el número esperado de descendientes que generó el mejor individuo de la población: $PS = \frac{f_{\text{máx}}}{\bar{f}}$. Es la relación entre el valor máximo de evaluación de toda la población y el valor medio.

Lo ideal son valores de presión selectiva alrededor de 1,5. Valores mayores generarán superindividuos, valores menores hacen la búsqueda más lenta. En caso extremo, si PS = 1, todos los individuos tienen idéntica evaluación y no existe ninguna presión selectiva

Para cuantificar la diversidad hay que calcular la frecuencia relativa y calcular su desviación típica:

$$fr_i = \frac{f_i}{\sum_{j=1}^{n} f_j} \quad D = \sqrt{\sum_{j=1}^{n} (fr_j - \bar{fr})^2}$$

Controlar el mecanismo de selección, introduciendo nuevos operadores (tipo torneos) que prevengan de superindividuos.

Cambiar la función de evaluación, de forma que suavice las diferencias entre los individuos.

Introducir mecanismo de control de edades, donde los individuos reducen su aptitud con la edad o simplemente desaparecen.

# 4. TEMA 3: TÉCNICAS DE COMPUTACIÓN EVOLUTIVA

## 4.1. Estrategias Evolutivas

### 4.1.1. Diferencias con los Algoritmos Genéticos

En lugar de tener codificación en binario, se tienen valores reales, no son valores discretos, sino continuos.

Al no ser discretos, no podemos hacer mutación discreta, en su lugar se hace mutación gaussiana N(0, $\sigma$).

En los algoritmos genéticos el elemento heurístico era cruce y en las estrategias evolutivas es la mutación.

#### Tipos

- Dependiendo del tamaño de la población.
- Por la forma en las que las nuevas generaciones son generadas.

Las poblaciones se mantienen en tamaño a lo largo de las generaciones.

- Si es con + se genera 1 individuo y hay 1 individuo en la población, de entre los 2 individuos se coge el más conveniente.
- Si es con , se genera 1 y hay 1, solo se considera el más nuevo.

### 4.1.2. (1+1)-EE

Es un método de escalada puro.

La población está compuesta por un solo individuo ($a_1^t$), que variara entre generaciones según: $P^t = (a_1^t) = ((x_1^t, \sigma_1^t))$.

El individuo está formado por dos vectores de números reales:

- Vector de la codificación $x^t$, codifica una solución al problema.
- Vector de varianzas $\sigma^t$, nos indica como de alejada se encuentra la solución de la solución óptima.

### Mutación

En cada generación t se produce una población intermedia de 2 individuos, el individuo nuevo es generado mediante mutación.

$$P'^t = (a_1^t, a_2^t) \in I \times I$$

$$a_2^t = m(P^t) = (x_2^t, \sigma_2^t))$$

El operador de mutación actúa de forma diferente con cada parte del individuo:

- Vector de codificación, se produce mutación mediante distribución gaussiana.

  A cada elemento del vector se le aplica una variación obtenida de forma aleatoria siguiendo una distribución gaussiana: $x_2^t = x_1^t + N_0(\sigma_2^t)$.

  La distribución normal generará valores cercanos al cero, tanto positivos como negativos, con una probabilidad tanto más grande cuanto menor sea el valor de $\sigma$. De esta manera se irán ajustando poco a poco.

  - Sigma pequeño, explotación.
  - Sigma grande, exploración.

  Se genera una nube gaussiana de probabilidad de encontrar el individuo en una determinada posición, después de la mutación la probabilidad se concentra en torno a la posición actual del individuo. Su dispersión viene dada por el valor de $\sigma$

- Vector de varianzas, se muta según la regla del 1/5.

  Cuanto más lejos de solución óptima es más probable que se produzca una mejora, y cuanto más cerca menos probable.

  Por lo que si se mejora de forma constante es que estoy lejos de la solución (fácilmente mejorable) y se hará la sigma más grande, cuando cuesta más hacer mejoras estoy cerca (difícilmente mejorable) y reduzco su valor.

### Regla del 1/5

Se almacena en $\psi_s^t$ la proporción de veces que la población ha cambiado en las últimas generaciones.

Se aplica:

$$\sigma^{t+n} = \begin{cases} c \cdot \sigma^t & si\ \psi_s^t < \frac{1}{5}\ reduce \\ \frac{\sigma^t}{c} & si\ \psi_s^t > \frac{1}{5}\ aumenta \\ \sigma^t & si\ \psi_s^t = \frac{1}{5}\ mantiene \end{cases}$$

Es necesario que $c < 1$. El valor se obtiene experimentalmente, pero se recomienda un valor en torno a 0,82.

Esta regla asegura la convergencia del método:

- Si las varianzas aumentan, se están obteniendo soluciones cada vez mejores y en algún momento se encontrará el óptimo
- Si se encuentra el óptimo, las siguientes soluciones serán peores, y la varianza decrecerá hasta cero
- Si no se encuentra el óptimo, las soluciones no mejorarán, y pasará lo mismo
- Cuando la varianza es cero, no se generan nuevas soluciones.

### Evolución

En la generación t+1, se evalúan los dos individuos de la población (1 previo y 1 nuevo) y se selecciona el mejor de los dos.

Por lo que si el nuevo es el mejor, la nueva población es el nuevo individuo, si no se mantienen la misma población.

$$P^{t+1} = s(P'^t) = \begin{cases} a_2^t & si\ f(x_1^t) \leq f(x_2^t) \\ a_1^t = P^t & en\ caso\ contrario \end{cases}$$

**Procedimiento** $(1 + 1) - EE$

1. Generar aleatoriamente un vector de números reales y sus varianzas. Las varianzas tomarán valores grandes.
2. Evaluar el individuo generado.
3. Repetir hasta cumplir criterio de convergencia:
   *a*) Generar una nueva solución a partir de la anterior mediante mutación, mutar el vector de codificación.
   *b*) Evaluar el individuo generado.
   *c*) Eliminar el individuo cuyo valor de adaptación sea menor.
   *d*) Si el individuo que queda es el nuevo, aumentar la frecuencia de éxitos, si no disminuirla.
   *e*) Mutar el vector de varianzas de acuerdo con la regla 1/5.
4. Producir como resultado el individuo resultante

### 4.1.3. EE Múltiples

Ahora no solo tendremos poblaciones de 1 solo individuo.

Ahora sí se podrá utilizar el operador de sobrecruzamiento, por lo que habrá que definir uno para vectores de números reales.

Como se tienen más individuos ahora no se podrá utilizar la mutación con reglas 1/5, ya no hay una estadística de mejora para cada individuo.

#### Cruce en EE

Actúa de forma separada sobre el vector de codificación y varianza.

Se realiza sobre un conjunto de p individuos, normalmente 2, 3 o 4 individuos, para generar 1 nuevo individuo.

- Vector de codificación, $\vec{x'} = \frac{1}{p}(\sum_{i=1}^{p} x_1^i, \sum_{i=1}^{p} x_2^i, ..., \sum_{i=1}^{p} x_\lambda^i)$, cada nuevo valor es la media de los valores de los progenitores.
- Vector de varianzas, se hace cruce posicional, tiene para cada varianza el valor de otro de los individuos aleatoriamente.
  $\vec{\sigma'} = (rnd(\sigma_1^1, \sigma_1^2, ..., \sigma_1^p), rnd(\sigma_2^1, \sigma_2^2, ..., \sigma_2^p), ..., rnd(\sigma_\lambda^1, \sigma_\lambda^2, ..., \sigma_\lambda^p))$

#### Mutación en EE

Vector de codificación, se hace de la misma manera, mediante distribución gaussiana, a cada elemento del vector se le aplica una variación obtenida de forma aleatoria siguiendo una distribución gaussiana: $x_2^t = x_1^t + N_0(\sigma_2^t)$.

Vector de varianzas, ya no se utiliza la regla de 1/5, sino que se va reduciendo poco a poco según un esquema Gaussiano. Se puede utilizar una de las siguientes:

- $\sigma' = \sigma \cdot e^{N(0,\tau)}$

  $\tau$ es la tasa de aprendizaje que toma un valor aproximado de $\tau \approx \frac{b}{\sqrt{2 \cdot \sqrt{\lambda}}}$, donde $\lambda$ es la longitud de los individuos.

- $\sigma' = e^{N(0,\tau_0)} \cdot \sigma e^{N(0,\tau)}$

  Este es otro método donde introduce el parámetro $\tau_0$ cuyo valor es $\tau_0 = \frac{b}{\sqrt{2 \cdot \lambda}}$

Se recomienda un valor de b aproximado de 1.

#### Evolución

Las poblaciones no tienen por qué tener el mismo tamaño, $\mu$ es el tamaño de la población y $\lambda$ es el tamaño de la población de nuevos individuos.

La selección se puede hacer por muestreo aleatorio simple (ruleta) o cualquier otro método similar.

Solapamiento: No se genera una población completa, los nuevos individuos sustituyen a algunos de los de la generación precedente.

Políticas de inserción y reemplazo: Hay que definir cómo se van a insertar los nuevos individuos y como se van a eliminar los sobrantes.

#### Políticas de inclusión y reemplazo

Si se generan $\lambda$ descendientes a partir de $\mu$ progenitores.

- Reemplazo por inclusión $(\mu + \lambda) - E$: Se unen las poblaciones y se seleccionan los $\mu$ mejores según la función de adaptación.

  Los padres como los descendientes compiten para ser incluidos.

- Reemplazo por inserción $(\mu, \lambda) - E$: Los descendientes siempre pasan a la siguiente generación y sustituyen a los menos aptos.

  - Si $\mu \leq \lambda$ se cogen de los $\lambda$ los $\mu$ mejores.
  - Si $\mu \geq \lambda$ se eliminan de los $\mu$ $\lambda$ individuos para poder introducir los $\lambda$ nuevos.

### 4.1.4. Procedimiento $(\mu, \lambda) - EE$

1. Generar aleatoriamente un conjunto de vectores de números reales y sus varianzas. Las varianzas tomarán valores grandes.
2. Repetir hasta cumplir criterio de convergencia:

   *a*) Evaluar la población generada.

   *b*) Ordenar la población de forma decreciente al fitness.

   *c*) Repetir $\lambda$ veces:

      1) Seleccionar 2, 3 o 4 individuos de forma proporcional a su fitness.
      2) Generar un nuevo individuo aplicando sobrecruzamiento, media para codificación y varianza aleatoria entre los progenitores.
      3) Mutar este último individuo.

*d*) Generar una población intermedia con los $\lambda$ descendientes generados en el paso anterior.

*e*) Eliminar de la población original los últimos $\lambda$ elementos, al estar ordenados.

*f*) Generar una nueva población mediante la unión de la población original y la intermedia.

3. Producir como resultado el mejor individuo de la población. resultante

#### 4.1.5. Procedimiento $(\mu + \lambda) - EE$

1. Generar aleatoriamente un conjunto de vectores de números reales y sus varianzas. Las varianzas tomarán valores grandes

2. Repetir hasta cumplir criterio de convergencia:

   *a*) Evaluar la población generada.

   *b*) Repetir $\lambda$ veces:

      1) Seleccionar t individuos de forma proporcional al fitness.
      2) Generar un nuevo individuo aplicando sobrecruzamiento.
      3) Mutar este último individuo.

   *c*) Añadir a la población todos los elementos generados en el paso anterior. El tamaño de la población será ahora $\mu + \lambda$.

   *d*) Eliminar de la población los peores $\lambda$ elementos.

3. Producir como resultado el mejor individuo de la población resultante.

### 4.2. Programación Genética

Los métodos se diferencian por la codificación, el resto es igual con pequeñas variaciones. En este caso se utilizará una codificación más cercana al problema real, más representativa.

#### 4.2.1. Orígenes

Técnica derivada de los AG ideada en 1989 para representar en LISP, en el que hay una estructura fija y se pueden poner operadores o terminales.

El sistema de representación es mucho más expresivo (mayor nivel de abstracción), pero permitiendo los operadores genéticos. Aunque es importante que mantenga su carácter de simbología discreta, es decir que haya un conjunto de elementos finitos entre los que elegir.

### 4.2.2. Representación de las soluciones

Los individuos se representan mediante expresiones, en sintaxis de Lisp.

Árboles cuyos nodos son funciones y las hojas operandos.

- Cada nodo (función) tiene tantas ramas como argumentos tiene la función que la compone (operandos).
- Un argumento puede ser a su vez otra función, por lo que se anidará, o un valor o variable, en cuyo caso no hay continuación (terminal).

Se acaba construyendo un árbol cuya ejecución es una solución a un problema.

Hay que definir el conjunto de funciones permitidas y su aridad (número de argumentos) y el de terminales.

### 4.2.3. Resolución de problemas

A la hora de resolver problemas se debe enriquecer la representación de forma que permita cualquier tipo de fórmula, con restricciones.

Se parte de una **hipótesis**, para hacernos la idea de que buscamos. En el caso de las ecuaciones de segundo grado es: La solución es una fórmula algebraica simple, es decir con operadores *, -, /, exponentes y raíces.

Lo primero en un problema de este tipo es definir que lo puede componer, los posibles valores. A partir de la hipótesis creamos los conjuntos de funciones y terminales.

Dichos conjuntos sirven de base para construir los árboles (soluciones).

Lo que no se incluya no podrá aparecer en las soluciones.

En el caso de las ecuaciones de segundo grado se asignan aleatoriamente valores, se resuelve la ecuación y con la x obtenida se sustituye en la fórmula base de las ecuaciones de segundo grado y debe dar 0.

### 4.2.4. Evaluación de un individuo

Un individuo debe representar una fórmula para resolver cualquier ecuación de ese tipo. Para evaluarlo hay que saber hasta qué punto es capaz de resolver cualquier ecuación de esa clase.

No se puede probar con todas las ecuaciones posibles, por lo que habrá que hacer una estimación de su eficacia probando con un conjunto finito de ecuaciones.

Las ecuaciones que nos servirán de evaluación se generan aleatoriamente y se calcula cómo resuelve el individuo dicho conjunto.

Se sustituyen dichos valores en la ecuación representada por el individuo a evaluar, y así se obtiene un valor para la x.

Se sustituye dicho valor en la ecuación de segundo grado, y se comprueba si cumple el criterio o no.

Si es así, el individuo resuelve la ecuación-ejemplo, si no es así, la diferencia se utilizará para el cálculo del fitness del individuo.

La evaluación final podría ser la suma de los valores obtenidos para cada ecuación ejemplo, la media o el máximo.

### 4.2.5. Procedimiento

Mientras no se cumpla el criterio de convergencia:

1. Ejecutar cada programa y calcular fitness.
2. Crear una población intermedia vacía
3. Copiar en ella los mejores individuos
4. Mientras la población intermedia no esté completa:
    *a*) Seleccionar un operador entre mutación y sobrecruzamiento
    *b*) Si se ha elegido mutación:
        1) Seleccionar un individuo aleatoriamente de la población original
        2) Crear un nuevo individuo por mutación del anterior
        3) Incluir el nuevo individuo en la población intermedia
    *c*) Si se ha elegido sobrecruzamiento:
        1) Seleccionar dos individuos aleatoriamente de la población original
        2) Crear dos nuevos individuos por sobrecruzamiento de los anteriores
        3) Incluir los nuevos individuos en la población intermedia
5. Sustituir a los individuos de la población por los individuos de la población intermedia

#### Procedimiento Población Inicial

Generar una población inicial de programas aleatoriamente:

1. Se asignan probabilidades a la elección de terminal y función respectivamente
2. Se selecciona entre terminal y función mediante las probabilidades anteriores

3. Si se ha seleccionado función se elige entre las del conjunto de funciones, y se crea un nodo del árbol

4. Si se ha seleccionado terminal se elige entre los símbolos del conjunto de terminales, y se crea una hoja del árbol. Si el símbolo es una constante, se genera un valor para esa constante de forma aleatoria en el rango especificado

### 4.2.6. Inicialización

Como se trata de generar estructuras en forma de árbol se debe determinar p, hasta que profundidad deberán llegar los individuos como máximo.

- **Método full:** Se generan en amplitud, se van eligiendo aleatoriamente para los nodos (no hojas) funciones hasta p-1 y cuando se llega a las hojas p se asignan terminales.

- **Método grow:** En cada nodo hay una probabilidad de que se genera una función o un terminal (no obliga a que solo sean terminales), pero llegada a la profundidad p se podrán poner solo terminales.

Para generar más variedad se puede generar la mitad de la población con full y la otra como grow, o también ir variando la profundidad.

### 4.2.7. Selección

Se realiza de la misma manera que en los Algoritmos Genéticos y en las Estrategias evolutivas, es decir se tendrá en cuanta la evaluación de los individuos para elegirlos para reproducir.

Los tipos son: Proporcional (ruleta), Jerárquico, Torneos.

### 4.2.8. Cruce

Consiste en la selección de un nodo aleatorio de cada individuo, que no sean la raíz o los terminales, solo operadores. Una vez seleccionado se intercambian los subárboles de los progenitores, dando lugar a dos nuevos individuos.

Es capaz de generar soluciones nuevas a partir de idénticas, esta es una capacidad muy buena para generar variedad. Aunque sufre de epístasis, una pequeña variación cambia mucho la solución.

### 4.2.9. Mutación

Consiste en generar un nuevo programa a partir de un único progenitor, en este caso hay 3 tipos:

- Mutación terminal simple, cambiar un terminal.
- Mutación funcional simple, cambiar una función.
- Mutación de árbol, sustituir un árbol por otro aleatorio.

#### Mutación terminal simple

Se selecciona aleatoriamente un símbolo terminal del individuo y se sustituye por otro diferente del conjunto de terminales posibles.

#### Mutación funcional simple

Se selecciona aleatoriamente una función dentro del individuo y se sustituye por otro diferente del conjunto de funciones posibles, pero deben tener la misma aridad (número de parámetros).

#### Mutación de árbol

Se selecciona un subárbol del individuo, como en cruce, se elimina ese subárbol y se genera en su lugar un subárbol aleatorio.

### 4.2.10. Conjunto terminales

Consiste en:

- Las entradas externas al programa, a las que llamamos normalmente **variables** (a, b y c en el ejemplo de las ecuaciones de segundo grado)
- **Funciones** que no tienen argumentos. Ejecutan una pieza de código y devuelven un valor, o producen un efecto. Por ejemplo: rnd(), gira_derecha(), distancia_a_muro()
- **Constantes**. Valores que se generan aleatoriamente, o mediante algún procedimiento externo al programa.
    - Puede ser deseable que se generen **a partir de un conjunto prefijado** de posibles valores o un procedimiento muy restrictivo (ephemeral random constant). Una vez generada la "ephemeral" permanece **invariable** en el individuo, a no ser que mute a otra constante o sub-árbol.

#### 4.2.11. Conjunto funciones

Lo determina la naturaleza del problema:

- Aritmético: +, *, /, -
- Matemático: sen, cos, exp
- Booleano: And, Or, Not
- Condicional: If, Then, Else
- Iterativo: For, Repeat
- ...

#### 4.2.12. Clausura

Se debe cumplir esta propiedad para que pueda funcionar la Programación Genética:

- Las funciones deben ser consistentes.
- Los programas deben ser seguros.
- Las primitivas deben ser suficientes.

#### 4.2.13. Tipado

**Consistencia**

Es necesaria para que las funciones anidadas se ejecuten correctamente.

- Las funciones deben devolver el mismo tipo de dato.
- Las funciones deben recibir el mismo tipo de dato que devuelven.
- Se puede hacer conversión de tipos. Si el tipo que manejamos es booleano, y una función devuelve un entero, podemos considerar la parte negativa como F y la positiva como V.

#### Seguridad

Es necesaria para que las funciones se ejecuten correctamente sin generar errores.

- Utilizar versiones protegidas de las funciones. Verificar posibles problemas (división por cero) y devolver valor por defecto para cada problema detectado.
- Penalizar el fitness cada vez que se genere un problema.
- Acciones por defecto para movimientos prohibidos. Ej. Mover cuando hay un muro enfrente.

#### Suficiencia

El conjunto de primitivas debe ser capaz de expresar las soluciones del problema.

- Un conjunto de primitivas suficiente para problemas booleanos sería: And, Not, Or, x1, x2 ....
- Uno insuficiente: +, -, *, /, x, cte..., ya que no permite generar funciones trascendentes (Exp, sin, ...).
- Puede no ser un problema, ya que se pueden generar soluciones suficientemente válidas, sin aumentar excesivamente el espacio de búsqueda.

### 4.2.14. Sintaxis

Se pueden evolucionar todo tipo de estructuras: puentes, circuitos, antenas, lentes, ...

### 4.2.15. Fitness

El fitness se puede medir de maneras diversas:

- Error entre salida obtenida y deseada.
- Tiempo (combustible, dinero) para llevar el sistema a un determinado objetivo.
- La eficacia con la que un sistema es capaz de detectar patrones o clasificarlos.
- La puntuación obtenida en un determinado juego.
- El ajuste de una estructura a los requerimientos del usuario.

#### 4.2.16. Parámetros

- **Objetivo**: Función que interpole conj. de puntos
- **Conjunto funciones**: +, -, /, *
- **Conjunto terminales**: x, Cte.
- **Fitness**: Error de interpolación
- **Selección**: Ruleta no elitista
- **Población inicial**: Mitad y mitad; prof. 2; 50 % terminales Ctes.
- **Otros parámetros**: Tamaño 4; 50 % cruce en árbol 25 % mutación en árbol; 25 % reproducción.

#### 4.2.17. Estructuras modulares

Se pueden determinar automáticamente ciertas estructuras para convertirlas en funciones, y añadirlas al conjunto de funciones.

# 5. TEMA 4: RESOLUCIÓN DE PROBLEMAS MEDIANTE TÉCNICAS EVOLUTIVAS

## 5.1. Resolución de problemas con múltiples soluciones

**Problema multimodal:** Problemas con muchos picos de buenas soluciones y en los que queremos una serie de buenas soluciones (estarán en picos distintos).

### 5.1.1. Compartición de fitness

Los AG tienden a converger hacia una única solución. Es importante mantener al máximo posible la diversidad genética.

Se debe intentar mantener "nichos" de individuos parecidos, pero diferentes de un nicho a otro.

Existen diferentes técnicas para mantener la diversidad genética y obtener más de una solución:

- Métodos de nichos.
- Métodos de sobrepoblación.
- Sobrepoblación determinista.
- Selección restrictiva de torneos.

### 5.1.2. Nichos ecológicos

La compartición modifica el espacio de búsqueda reduciendo el beneficio en regiones más densamente pobladas.

Reduce el fitness de un individuo en una cantidad proporcional al número de individuos similares a él en la población: $f_i' = \frac{f_i}{m_i}$.

$m_i$ es una medida de la similitud entre el individuo i y el resto de los individuos de la población y se calcula mediante: $m_i = \sum_{j=1}^{N} sh(d_{ij})$, donde $sh$ es el share.

N es el tamaño de la población, $d_{ij}$ es la distancia entre i y j y sh($d_{ij}$) mide la similitud entre i y j.

Devolverá uno si los dos individuos son iguales y cero si su diferencia es mayor que un cierto umbral $\sigma_s$:

$sh(d_{ij}) = 1 - \left(\frac{d_{ij}}{\sigma_s}\right)^{\alpha}$ si $d < \sigma_s$ y es 0 en el caso contrario.

- $\alpha$ es el factor.
- $\sigma_s$ es el umbral, un valor entre 0 y 1, dependerá del dominio, pero es el mismo para todos los individuos (presupone picos equidistantes). No es posible asignarlo en todos los problemas.

### Nichos

Un buen individuo se convierte en menos buenos, si en la población existen un gran número de individuos parecidos a él. Un mal individuo, si está aislado, verá incrementado su valor de fitness.

De esta forma se mantienen zonas independientes de buenos individuos, pero poco abundantes. Esto permite mantener la variabilidad genética y converger a más de una solución, siempre que todas ellas sean lo suficientemente buenas.

### Distancia

La distancia puede ser:

- Genotípica. Se mide la distancia de la codificación de dos individuos. Si la codificación es binaria se suele utilizar hamming.
- Fenotípica. Se mide lo que se parecen las soluciones generadas por cada uno de los individuos a comparar. Depende del dominio del problema suele ser la euclídea.

  Suele dar mejores rendimientos en los métodos de nichos

Favorece la búsqueda en regiones inexploradas del espacio y favorece la formación de subpoblaciones estables

### Escalado del fitness

Para mejorar la eficacia de los nichos se puede escalar el valor de fitness $f_i' = \frac{f_i^{\beta}}{m_i}$.

El escalado incrementa la diferenciación entre óptimos y reduce el problema de la decepción (dos individuos buenos producen un individuo malo).

Si el valor de $\beta$ es demasiado alto, se pueden producir superindividuos que hagan converger prematuramente al método y si el valor de $\beta$ es demasiado bajo la diferenciación entre los óptimos puede ser insuficiente y no ser posible de detectar.

Se puede hacer un buen balance entre explotación y exploración variando el valor de $\beta$ a lo largo del tiempo. Por ejemplo mediante “simulating annealing.

### 5.1.3. Sobrepoblación

Se mantiene la diversidad haciendo que los nuevos individuos sustituyan a individuos similares.

Se utiliza un esquema de estado estacionario en el que se sustituye solo parte de la población en cada generación.

Se generan n individuos nuevos ($n < N$). Se seleccionan m individuos de la población de forma aleatoria $n < m < N$ o puede ser de toda la población.

De entre los m individuos se eliminan los n más parecidos a los n nuevos generados.

### 5.1.4. Sobrepoblación determinista

Se introduce competición entre progenitores y descendientes por los mismos nichos.

Cada descendiente reemplaza al progenitor más cercano (se utiliza la distancia fenotípica), pero solo si tiene mejor fitness que él.

Después del cruce se generan dos conjuntos:

- El padre 1 compite con el hijo 1, el padre 2 compite con el hijo 2.
- El padre 1 compite con el hijo 2, el padre 2 compite con el hijo 1.

Se escoge el conjunto con menor distancia entre elementos, padreX-hijoY+padreZ-hijoW. De entre los individuos de las 2 parejas padre-hijo, pasarán a la siguiente generación el de mayor fitness de cada pareja.

### 5.1.5. Selección de torneos restringida

Es una mezcla entre sobrepoblación y torneos.

Se seleccionan dos individuos mediante el operador de selección, se generan dos nuevos individuos mediante cruce y mutación, y se seleccionan aleatoriamente m individuos de la población. Cada descendiente compite con el individuo de la subpoblación anterior que más se le parece. Los ganadores pasan a la siguiente generación

Igual que Superpoblación, pero los descendientes no siempre pasan a la nueva generación, solo si son mejores que los individuos a los que van a sustituir.

Se generan n nuevos y todos ellos se comparan con el más parecido de una subpoblación de la población y entre esos 2 pasa el de mejor evaluación/fitness.

En todos estos métodos buscamos soluciones distribuidas.

## 5.2. Resolución de problemas con restricciones

En algunos problemas las soluciones deben cumplir una serie de criterios para ser válidas, en estos casos no todos los individuos codificados son aceptables. Un ejemplo es asignar tareas a empleados, cuando no todos los empleados son capaces de hacer todas las tareas.

### 5.2.1. Problemática

Los AG canónicos no pueden tratar restricciones.

Se puede modificar codificando un superconjunto X del conjunto de soluciones factibles $\mathcal{X}$ ($X \supseteq \mathcal{X}$)

El superconjunto debe: Ser fácil de codificar y no ser mucho más grande que el conjunto de soluciones factibles (para no perder excesivo tiempo).

### 5.2.2. Penalización

Las restricciones se suelen expresar como un conjunto de inecuaciones: $g_i(x) \geq 0$.

Para tratarlas se puede incluir un término de penalización en la función de evaluación que penalice más o menos en función del grado de incumplimiento de la restricción.

Matemáticamente podría definirse un grado de incumplimiento por parte del individuo x de la restricción i-ésima como:

$$Ir_i(x) = \begin{cases} -g_i(x) & Sii \quad g_i(x) < 0 \quad incumple \\ 0 & Sii \quad g_i(x) \geq 0 \quad no\ incumple \end{cases}$$

Este incumplimiento se introduce como penalización en la función de evaluación de cada individuo:

$$F(x) = F(x) - k \cdot \mathcal{P}(Ir_1(x), ..., Ir_p(x))$$

- p es el número de restricciones.
- k coeficiente de penalización.
- $\mathcal{P}$ es una función de penalización creciente. positiva dependiente del dominio.

### 5.2.3. Penalización variable

Se empieza con valores bajos, ya que al principio se incumplirán más restricciones, pero a medida que se progresa es menos permisivo.

Se pueden proponer funciones de penalización variables con el tiempo, cuanto más avanza más rígida:

$$P = \left(\frac{k \cdot t}{m}\right)^2 \frac{\sum_{i=1}^{p} Ir_i(x(t))}{F(P(t))}$$

#### Problemática de la Penalización

Las funciones de penalización muy severas no son recomendables en problemas fuertemente restringidos.

- El AG gasta mucho tiempo en procesar individuos no factibles.
- Cuando se encuentra un individuo factible este puede destacar demasiado sobre los demás (convergencia prematura).

Las funciones de penalización muy flexibles tienen el riesgo de dejar prosperar a individuos que, aun siendo no factibles, tengan una aptitud neta mayor que otros factibles.

A veces se habla de técnicas de gratificación: premian a los individuos factibles.

### 5.2.4. Reparación o Corrección

Consisten en habilitar un procedimiento con el que corregir cualquier solución no factible que se genere.

El inconveniente es que suele ser difícil encontrar un procedimiento de corrección lo suficientemente general y, si se encuentra, normalmente consume muchos recursos de computación.

Este procedimiento es necesariamente específico para cada problema.

### 5.2.5. Decodificación

Consisten en realizar la codificación de tal modo que sea imposible generar soluciones no factibles. Se modifica X a través de la codificación para que se aproxime lo máximo posible a $\mathcal{X}$.

Suelen requerir modificar los operadores genéticos para que conserven la factibilidad de los individuos.

Tiene inconvenientes similares a las técnicas de reparación. De modo indirecto todas estas técnicas están añadiendo conocimiento específico.

## 5.3. Resolución de problemas con varios objetivos contrapuestos

Los problemas de optimización multi-objetivo son aquellos que suelen tener dos o más funciones objetivo que deben satisfacerse simultáneamente y que posiblemente están en conflicto entre sí. (como seria calidad y precio en la compra)

No existe una solución mejor, sino un conjunto de alternativas, que suelen representar los mejores compromisos entre los diferentes objetivos, elimina los no óptimos.

Es tarea del usuario decidir qué solución es la más adecuada

En muchos casos es necesario disponer de todo el conjunto de soluciones.

Se introduce el término dominancia, que es cuando una solución es mejor en todos los objetivos que otra, la domina.

El Frente de Pareto está formado por las soluciones no dominadas, estas formarán las mejores soluciones.

### 5.3.1. Ejemplos

#### Manets - Mobile Ad-hoc Networks

Red de conexión inalámbrica, cada terminal se puede comunicar directamente con otras estaciones. El objetivo es optimizar la transmisión de mensajes por la red. Problema con tres objetivos: maximizar el número de estaciones alcanzadas minimizando el tráfico de la red y el tiempo de establecimiento de la comunicación.

#### RND - Radio Network Design

Determinar la localización de las antenas, con el objetivo de colocar el mínimo número de antenas posibles con la máxima cobertura.

#### Optimización de carteras de inversión

Encontrar una composición óptima de inversión en activos que nos produzcan el máximo rendimiento al menor riesgo.

**Diseño óptimo de estructuras reticulares**

En este caso los objetivos son: minimizar el peso de la estructura, minimizar los desplazamientos en cada nodo y minimizar la vibración de la estructura.

### 5.3.2. Solución mono-objetivo

Tradicionalmente, estos problemas se abordan:

- Modelando los problemas como monoobjetivo (ponderando los diferentes objetivos)
- Considerando algunos de los objetivos como restricciones al problema:
  - Penalización de individuos (fitness), es muy importante que esté balanceado.
  - Corrección de individuos
- Métodos de ejecución secuencial, se puede introducir nichos y sobrepoblación.
- Este solo obtiene una solución, la que maximice o minimice la función de fitness, los resultados no son muy buenos.

### 5.3.3. Definiciones

Un problema de optimización multiobjetivo general puede formularse como:

- Encontrar el vector $x = (x_1, x_2, ..., x_n)$ que satisfaga las m restricciones de desigualdad $e(x) = (e_1(x), e_2(x), ..., e_m(x)) \geq 0$ y que optimice $y = f(x) = (f_1(x), f_2(x), ..., f_k(x))$.

La noción más aceptada de «óptimo» en el entorno de problemas multiobjetivo es la propuesta por Pareto en 1896.

Se dice que un punto $x^*$ es un óptimo de Pareto (minimización) si para todo x del espacio de entrada se cumple: $f_i(x^*) \leq f_i(x) \forall i = 1, ..., k$.

**Conjunto de óptimos de Pareto.** P* está formado por los puntos x*, tales que no existe otro vector x tal que f(x) domine a f(x*).

**Frente de Pareto.** es el conjunto de vectores objetivos Y= f(P*).

### 5.3.4. Frente de Pareto

Cada punto del frente se corresponde con un individuo.

No hay una única solución. El método debe converger a un conjunto de soluciones (método de nichos).

Para elegir una solución, hay que establecer un compromiso entre los diferentes objetivos.

### 5.3.5. Primeros pasos

Dentro de la disciplina "Investigación operativa" se han estudiado técnicas para la resolución de problemas de optimización multiobjetivo.

La mayoría de ellas están limitadas a frentes de Pareto con ciertas características (p. ej. convexos) y suelen requerir un punto inicial de búsqueda.

Algoritmos evolutivos mono-objetivos, con restricciones o ponderando los diferentes objetivos. Solo se consigue una solución y no un frente de soluciones

Ejecutar los algoritmos mono-objetivo varias veces cambiando las restricciones o la ponderación: se obtiene un conjunto de soluciones

Rosenberg en 1967 ya habló del potencial de la AG para abordar problemas MO (MOEA).

### 5.3.6. MOEA's

Históricamente se considera que ha habido dos generaciones de algoritmos evolutivos para optimización multiobjetivo (MOEA):

- **Primera generación:** Caracterizada por el uso de jerarquización de Pareto y nichos. Algoritmos relativamente simples. Gozaron de éxito, pero comenzaron a decaer (NSGA, NPGA, MOGA y VEGA).
- **Segunda generación:** Transcurre desde 1999 a la actualidad. Hoy día son vistos como el estado del arte: SPEA, SPEA2, NSGA II, MOMGA, MOMGA II, PAES, PESA, PESA II.

La tendencia actual es extender otras heurísticas a problemas multiobjetivo: búsqueda dispersa, optimización mediante enjambres partículas, sistemas inmunes artificiales.

#### MOEA's segunda generación

Fundamentalmente, consiste en la incorporación de elitismo.

Se propusieron dos mecanismos para el elitismo:

- Mediante una selección ($\mu + \lambda$):
  - Dados μ padres, se generan $\lambda$ hijos.
  - Se crea una población de padres e hijos ($\mu + \lambda$).
  - Compiten padre e hijos pasando a la siguiente población los $\mu$ mejores.

- Mediante la utilización de una población externa (que almacena soluciones no dominadas). Algunos aspectos a considerar en este enfoque son:
  - La interacción entre la población y la población externa.
  - Los criterios para realizar las inserciones de elementos en la población externa.
  - Las acciones a tomar cuando se completa la población externa.

### 5.3.7. VEGA

El primer algoritmo evolutivo MO fue "Vector Evaluated Genetic Algorithm (VEGA)" desarrollado por Schaffer en 1984.

Para M objetivos, se divide aleatoriamente la población, en cada iteración, en M subpoblaciones del mismo tamaño.

A cada subpoblación se le aplica como función de evaluación un objetivo diferente.

Se aplica el operador de cruce a individuos que pueden pertenecer a subpoblaciones diferentes, generando soluciones que lo hacen bien con diferentes objetivos.

Es un AG canónico, en el que la evaluación de los individuos se realiza de forma aleatoria utilizando un objetivo u otro.

Es simple y fácil de implementar

Al generarse soluciones de compromiso entre objetivos, se pierde eficacia, ya que no derivan hacia el frente Pareto, sino a soluciones medias.

### 5.3.8. NSGA II

El algoritmo NSGA II (Elitist Non-Dominated Sorting Genetic Algorithm) fue propuesto por Deb et al. en 2000.

Es el más conocido y ampliamente utilizado

Se genera una población de descendientes $Q_t$ (tamaño N) a partir de la población de padres $P_t$ (tamaño N)

Se unen en una población intermedia de tamaño 2N ($R_t$)

Se ordena la población $R_t$ con el criterio de dominancia, obteniendo diferentes frentes

F1 son las soluciones no dominadas,

F2 son las no dominadas cuando hemos eliminado F1, etc.

La nueva población se construye a partir del primer frente (F1), si no está completa se añade el segundo(F2) y así sucesivamente.

Se buscan los mejores frentes, prefiriendo aquellos con los puntos más alejados cubriendo más soluciones.

La población descendiente $Q_i$ se crea a partir de la original $P_i$, usando selección por torneo, reproducción y mutación

Como la población $R_t$ es de tamaño 2N, y solamente N configuraciones pasarán a la siguiente generación. Se introducen hasta completar el tamaño N.

Si en un frente hay más individuos que los que quedan para llenar la población, se eligen a los más dispersos.

Se aplica un estimador de densidad conocido como "Crowding Distance", promoviendo la diversidad.

Elitismo, las dos mejores soluciones de la población actual se copian en la nueva.

"Crowding distance" de una solución del frente es la suma de las longitudes del lado del cubo formado por sus vecinos del frente Pareto.

Es un buen medidor de la diversidad cuando el número de objetivos es menor o igual que tres.

Para problemas con más de tres objetivos, no proporciona buenos resultados.

**Procedimiento NSGA II**

Combinar las poblaciones de padres e hijos ($R_t = R_t \cup Q_t$)

Calcular los frentes no dominados de manera jerárquica (F = NoD($R_t$))

Mientras la población no esté llena ($|P_{t+1}| + |F_i| <$N)

- Calcular la distancia "crowding" en $F_i$
- Incluir el i-ésimo frente no dominado en la población ($P_{t+1} = P_{t+1} \cup F_i$)
- Aumentar el contador de frentes (i = i + 1)

Ordenar los elementos del frente i-ésimo de manera descendiente

Elegir los elementos que quedan para completar la población entre los primeros de la ordenación anterior

Crear una población nueva mediante selección, cruce y mutación

Aumentar el contador de generaciones

### 5.3.9. SPEA

Strength Pareto Evolutionary Algorithm (SPEA) fue propuesto por Zitzler y Thiele en 1998.

Utiliza una población externa de soluciones no dominadas obtenidas previamente.

Se basa en el concepto de fuerza (strength), similar al rango de dominancia en otros MOEA. La fuerza de un individuo es proporcional al número de individuos que son dominados por él.

La asignación de fitness se basa en la medida de "fuerza".

Se utiliza clustering para mantener la diversidad.

En cada generación, los individuos no-dominados tanto de la población original como del archivo se utilizan para actualizar el archivo.

**Procedimiento SPEA**

1. Se inicializa la población y el salón de la fama (individuos no dominados)
2. Se calcula el fitness
3. Los individuos no dominados de la población y el salón de la fama pasan al nuevo salón de la fama (t+1)
    - Si hay de más se eliminan los excedentes mediante el operador de truncado
    - Si hay de menos se completa mediante la selección del entorno
4. Se seleccionan para el mating pool individuos del salón de la fama mediante torneos binarios, se selecciona un individuo del salón de la fama y otro que no para que compitan.
5. Se genera una nueva población mediante recombinación y mutación
6. Si no se cumple el criterio de parada vamos al paso 2

### 5.3.10. SPEA 2

SPEA-2 es una mejora al algoritmo original propuesta por Zitzler y Thiele en 2001.

Presenta tres diferencias sustanciales con SPEA:

- La asignación de fitness considera la cantidad de individuos dominados por cada solución (como SPEA), pero además toma en cuenta la cantidad de individuos que la dominan.
- Para mantener diversidad utiliza una técnica de estimación de densidad de individuos vecinos.
- Utiliza un esquema de truncamiento de la población externa que le garantiza la preservación de soluciones de frontera (bordes).
- Si el número de individuos no dominados es mayor que el tamaño de la población, se aplica un operador de truncamiento basado en el cálculo de las distancias a los k-vecinos más cercanos.

#### SPEA 2 Evaluación

A cada individuo se le asigna un valor de fuerza que es el número de individuos, entre población y fama, que domina $S(i) = |\{j, , j \in P_t + \overline{P}_t \wedge i \succ j\}|$

El fitness se calcula como la suma de las fuerzas de los individuos a los que domina $R(i) = \sum_{j \in P_t + \overline{P}_t, j > i} S(j)$

Se calcula para cada individuo un valor de densidad que determina lo aislado que está dicho individuo, mediante su distancia a su k-ésimo vecino más próximo ($\sigma_i^k$): $D(i) = \frac{1}{\sigma_i^k + 2}$

k se calcula a partir del tamaño de la población N y el tamaño del salón de la fama $\overline{N}$: $k = \sqrt{N + \overline{N}}$

El fitness definitivo es la suma del anterior y la densidad, $F(i) = R(i) + D(i)$

#### SPEA 2 Selección del entorno

Si el salón de la fama es demasiado pequeño, se añaden de entre los dominados aquellos con mayor fitness hasta que sea necesario.

Si el salón de la fama es demasiado grande se eliminan, de entre los no dominados, aquellos que estén más poblados, es decir con menor distancia a su k vecino más cercano.

### 5.3.11. Métricas

Cuando se resuelve un problema Multiobjetivo:

- El objetivo es obtener el frente de Pareto
- Hacerlo con rapidez: eficiencia
- Frente de calidad: convergencia y diversidad

En mono-objetivo, el mejor algoritmo será el que calcule la solución con mejor (óptimo) valor de la función objetivo.

En multi-objetivo, el mejor algoritmo será el que calcule el mejor frente de Pareto ¿Cómo medir si un frente es mejor que otro?

Las métricas deben evaluar:

- La calidad de los resultados (cercanía al frente de Pareto).
- La diversidad de las soluciones
- La cantidad de elementos del conjunto de óptimos de Pareto.
- La eficiencia computacional del algoritmo

Medidas de cardinalidad:

- La cardinalidad de un frente es el número de soluciones que existen en dicho frente
- Se prefieren frentes con más soluciones

Medidas de precisión:

- Miden la convergencia del frente obtenido respecto de un teórico frente óptimo
- Se calcula la distancia de nuestro frente al frente ideal
- Si el frente ideal es desconocido, se utiliza un frente de referencia R en su lugar

Medidas de diversidad

- Miden la distribución de los puntos del frente Pareto
- Distribución. Distancia relativa entre los puntos del frente. Mide los dispersos que están los puntos
- Difusión. Rango de valores cubiertos por la solución del frente. Mide lo amplio que es el frente

### Número de soluciones

Número de soluciones del frente

- Fácil de calcular.
- No refleja la calidad del frente .

Número de soluciones óptimas del frente

- Fácil de calcular.
- Requiere conocer el frente óptimo.
- No refleja la distribución de las soluciones.

### Hipervolumen

Mide el volumen de la región dominada por el frente $HV(A) = \cup_{i=1...|A|} v_i$ donde $V_i$ es el hipercubo con esquinas $a_i$ y W.

Sirve para medir la convergencia y la diversidad

No requiere conocer el frente óptimo

Sí se necesita un punto de referencia

Una de las métricas más utilizadas

A mayor HV, más diverso es el frente

### Spread

Es una medida de la distribución de los puntos contenidos en un frente

$$SP(A) = \frac{(d_L + d_U + \sum_{i=1}^{N-1} |d_i - d|)}{d_L + d_U + (N-1)d}$$

$$d = \frac{(d_1 + d_2 + ... + d_{N-1})}{N-1}$$

Válida solo para 2 objetivos

### Set coverage

Es una medida de la cobertura (en términos del operador de dominancia) de un frente sobre otro

$$SC(x, y) = \frac{|y \in Y, \exists x \in X, y \leq x|}{|Y|}$$

- Es una medida de comparación de dos frentes.
- No requiere conocer el frente Pareto óptimo.
- Siempre es conveniente evaluar SC(X, Y) y SC(Y, X).
- No se da información sobre la diversidad.
- No índica cómo de alejado están los frentes.

### Distancia generacional

Distancia media de cada solución en un frente hacia el frente de Pareto óptimo.

- Se puede usar con distintos tipos de distancia.
- Requiere conocer el frente óptimo.
- Un frente con un solo punto, sería una buena solución con esta métrica.

$$GD(A) = \frac{\sum_{i...n} d}{n}$$

En muchas ocasiones el frente óptimo se desconoce. Cuando se pretende comparar diferentes algoritmos, se obtiene un "frente óptimo" mezclando todas las soluciones y obteniendo las soluciones no dominadas. Este conjunto podría ser el frente de referencia

### Punto rodilla

Es un subconjunto de soluciones de un frente Pareto, en las cuales una mejora en un objetivo produce una severa degradación de al menos otro de los objetivos.

Soluciones del punto rodilla de un frente Pareto son preferibles si no se especifica ninguna otra preferencia

## 5.4. Algoritmos Coevolución

### 5.4.1. Diploidad

Se basa en que en algunos organismos, los cromosomas se encuentran duplicados, por lo que posee el fenotipo repetido.

Existen dos genes, llamados homólogos, para el mismo carácter

Si genes homólogos contienen el mismo valor se llaman homocigóticos

Si genes homólogos contienen distintos valores se llaman heterocigóticos

El carácter viene determinado por esquema de dominancias:

- Si homocigóticos codifican el mismo carácter y no hay problema.

  Color de ojos(azul, azul) = azul

- Si heterocigóticos, el que domina sobre el otro es el que produce el carácter.

  Color de ojos(azul, marrón) = marrón

#### Modelos computacionales diploides

Esquema triádico de Hollstien-Holland: Alelos=0, $1_0$, 1

- El $1_0$ es un 1 recesivo, y el 1 es dominante
- El 1 domina sobre los otros dos
- El 0 domina sobre el $1_0$

| | **0** | **$1_0$** | **1** |
|---|---|---|---|
| **0** | 0 | 0 | 1 |
| **$1_0$** | 0 | 1 | 1 |
| **1** | 1 | 1 | 1 |

Tabla 5.1: Esquema triádico de Hollstien-Holland

**Esquemas de dominancia**

División de Brindle:

- **Dominancia global, fija y aleatoria**
  - La dominancia de alelos binaria es determinada para cada locus
  - Al principio se crea un mapa de dominancia aleatoriamente por locus
  - La dominancia se mantiene durante todo el proceso
- **Dominancia global variable:** Asigna una probabilidad de dominar según su frecuencia, una vez se asigna toda la generación se sigue ese esquema, se asigna cada generación cuál será la dominancia.
  - La probabilidad de dominancia de un alelo en un locus es igual a la tasa de alelos en ese locus en la población
  - Se calcula cada generación mediante ensayos de Bernoulli
- **Dominancia global, determinista y variable:** Domina en cada alelo el valor más frecuente en la población.
  - Se calcula para cada locus la proporción de sus alelos
  - El alelo dominante es el más frecuente
  - Se reasignan dominancias tras cada generación
- **Se elige un cromosoma al azar**
  - Se elige un cromosoma aleatoriamente
  - Todos sus alelos se convierten en dominantes en sus respectivos locus
- **Dominancia del mejor cromosoma**
  - Se elige siempre el mejor cromosoma como dominante
- **Dominancia adaptativa**
  - Se incluye un nuevo cromosoma que no codifica una solución a un problema, sino el esquema de dominancias a utilizar.
  - De esta forma el esquema de dominancias puede evolucionar al igual que las soluciones.
  - Dar lugar a esquemas más eficaces.

### 5.4.2. Reproducción de individuos diploides

Se generan dos cigotos de cada individuo mediante sobrecruzamiento, se parten los dos cromosomas de los dos individuos y se cruzan dentro del mismo individuo.

Entonces se intercambian los cigotos de los dos individuos y tenemos 2 individuos haploides nuevos.

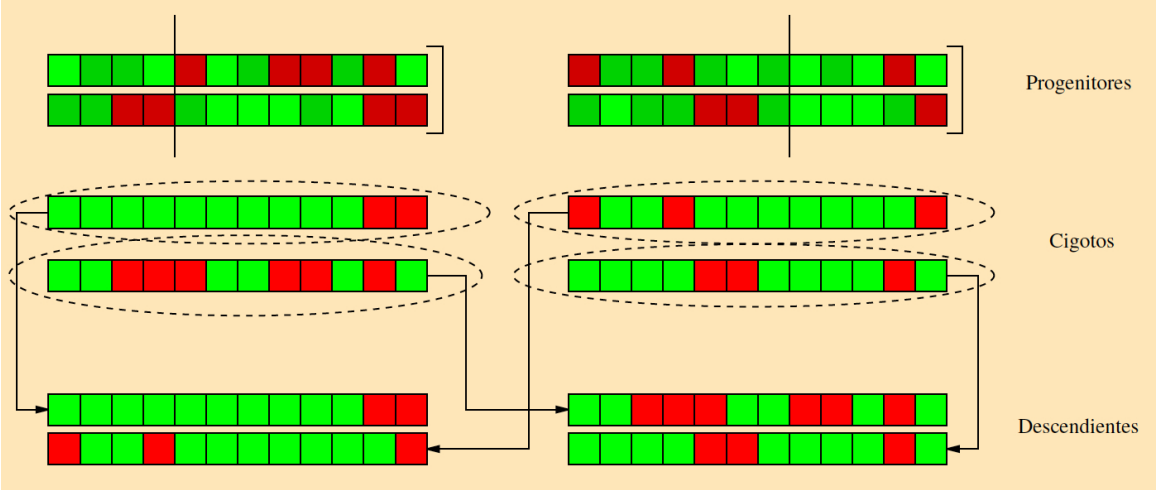



Fig. 5.1: Reproducción de individuos diploides

### 5.4.3. Coevolución

En lugar de haber un sistema en el entorno hay varios que compiten por los recursos, por lo que los sistemas compiten y tratan de ser superior a otro para sobrevivir.

La coevolución es una fuerza importante hacia la generación de adaptaciones muy complejas.

Relaciones depredador-presa

Existe una gran presión para que las presas se defiendan mejor y en respuesta los depredadores desarrollan mejores estrategias de ataque.

El éxito de una parte es el fracaso de la opuesta que debe desarrollar respuestas para mantener sus posibilidades de supervivencia, el dilema de la Reina roja. Para seguir siendo igual de bueno hay que seguir evolucionando, si no el resto lo harán y nosotros empeoraremos.

Normalmente se habla de 2 sistemas.

### Coevolución computacional

**Tipos:**

- **Cooperativa:** Dos sistemas independientes cooperan para aumentar sus posibilidades de supervivencia

  Simbiosis, "vive y deja vivir" en la Primera Guerra Mundial

- **Competitiva:** Dos sistemas independientes compiten por recursos compartidos

  Depredador-presa

  f1 inversamente proporcional a f2, si S1 mejora S2 tendrá mayor presión para evolucionar.

### Evaluación de los individuos

En algunas ocasiones los individuos de una población tienen que ser evaluados utilizando ejemplos. Hay que generar un conjunto de ejemplos, p. ej. de forma aleatoria, y utilizarlos para evaluar un individuo.

Aparece el problema denominado **"Ideal trainer", entrenador ideal**:

- Existe un número enorme de posibles ejemplos, pero solo un ínfimo número de ellos se pueden utilizar para evaluar al individuo para que el proceso sea eficiente. El problema está en cómo determinar qué conjunto de n ejemplos (n pequeño) es el "ideal" para que sirva de evaluación.
- Si el conjunto de ejemplos es "fácil" se puede producir especialización. Lo que puede ser un buen conjunto de evaluación para un individuo, puede ser malo para otro y viceversa

### Problema del entrenador ideal

Una primera solución sería generar un **nuevo conjunto de entrenamiento en cada generación**

- Los individuos no podrían especializarse, ya que se evalúan cada vez con ejemplos diferentes.
- Un individuo muy bueno en una generación, y que por tanto ha engendrado muchos descendientes, puede ser muy malo en la siguiente, y todos los individuos que ha engendrado ser inoperativos.
- Al cambiar bruscamente la evaluación de los individuos el "fitness landscape" se hace muy abrupto, y el método se desorienta.

Se podría **cambiar de conjunto de aprendizaje cada "n" generaciones**. Esto reduciría los problemas anteriores, pero no los eliminaría.

Se podría **cambiar "ligeramente" el conjunto de aprendizaje cada "n" generaciones**

- Habría que ver cuánto y cómo se cambia el conjunto de aprendizaje, y en función de qué.
- Al utilizar el mismo conjunto de aprendizaje para todos los individuos, cambios positivos para unos podrían ser negativos para otros.

Se podría generar un **conjunto de aprendizaje diferente para cada individuo, y cambiarlo "ligeramente" cada "n" generaciones**

- Se tendría el problema de cuánto y cómo cambiar el conjunto de aprendizaje.
- Si el conjunto de aprendizaje es complejo, el problema anterior no es trivial.

### Conjuntos de entrenamiento como sistemas genéticos

La solución es generar una **población de conjuntos de entrenamiento, cada individuo es un conjunto de entrenamiento**, compuesto por n valores reales.

Cada individuo-conjunto-de-entrenamiento es evaluado con un conjunto de individuos solución.

El resultado de la evaluación se utiliza para evaluar a los individuos-ejemplo y a los individuos-solución, una buena evaluación para un individuo-solución será mala para un individuo-ejemplo y viceversa.

Esta evaluación se utiliza dentro de un Algoritmo Genético, de forma que se evolucionan los conjuntos de entrenamiento más complicados para un individuo-solución o para un conjunto de individuos-solución.

### Life-time fitness

Dos poblaciones, una de soluciones $P_s(t)$ de tamaño $||P_s||$ y otra de ejemplos $P_e$ de tamaño $||P_e||$.

**$P_s(t)$ evoluciona con el tiempo, $P_e$ se mantiene fija.** Cada individuo de $P_s$ se evalúa sobre los últimos $\tau$ ejemplos, y lo mismo cada individuo de $P_e$.

La evaluación de $P_s$ sirve para seleccionar y generar nuevos individuos, mientras que la de $P_e$ solo para seleccionar individuos. La evaluación es la suma, u otra medida estadística, de encuentros entre soluciones y ejemplos.

**Ciclo coevolutivo**

1. Realizar n veces
   *a*) s = seleccionar($P_s$)
   *b*) e = seleccionar($P_e$)
   *c*) f = encuentro(s, e)
   *d*) actualizar-fit(s, f) se puede hacer con la media de los fitness
   *e*) act-fitness(e,1/f) se puede hacer con la media de los fitness
2. $s_1$ = seleccionar($P_s$)
3. $s_2$ = seleccionar($P_s$)
4. $s_{hijo}$ = cruce-mut($s_1$, $s_2$)
5. Realizar m veces
   *a*) e = seleccionar($P_e$)
   *b*) f = encuentro($s_{hijo}$, e)
   *c*) f($s_{hijo}$) = actualizar-fit($s_{hijo}$, f)
   *d*) f(e) = actualizar-fit(e,1/f)
6. insertar($s_{hijo}$, f($s_{hijo}$), $P_s$)

**Funciones**

- **Seleccionar(P)**. selecciona un elemento de la población (P), de manera proporcional a su fitness.
- **Actualizar fitness(i, f)**. Cada individuo (i) lleva una memoria del resultado de los últimos m encuentros, al actualizar, se elimina de la memoria el resultado del encuentro más antiguo, se añade el actual (f), y se recalcula su fitness.
- **Encuentro(s, e)**. La solución s se evalúa con el ejemplo e y obtenemos un valor de fitness para este encuentro.
- **Insertar(s, f(s), P)**. Si f(s) es mejor que el fitness del peor individuo de la población.

**Características**

- Las soluciones solo se evalúan con un conjunto pequeño (m) de ejemplos, en vez de con todos los posibles.
- Al principio todos los ejemplos tienen la misma probabilidad de ser seleccionados, pero a medida que son evaluados unos tienen más probabilidad que otros.

- Los **ejemplos más difíciles tendrán mejor fitness y mayor probabilidad de ser elegidos**.
- Al utilizar una ventana temporal de m para el cálculo del fitness, tanto soluciones como ejemplos se adaptarán unos a otros.