



# Programación Evolutiva

Tema 8: Extensiones de los algoritmos genéticos. Otros temas

Carlos Cervigón, Lourdes Araujo. 2023-2024

# Extensiones a los algoritmos genéticos

- ❑ Extensiones del modelo básico de Algoritmos Genéticos.
- ❑ Conservan las ideas fundamentales de los AGs.
- ❑ Introducen cambios en la estructura básica.
  - Algoritmos Genéticos con edades
  - Búsqueda Multiobjetivo
  - Algoritmos Genéticos Paralelos
  - Algoritmos Genéticos Diploides
  - Algoritmos meméticos
  - PSO, ACO, CRO
  - Arte Evolutivo

# Algoritmos Genéticos con edades

- ❑ Al crear un individuo
  - Se le asigna una duración, que depende de su aptitud.
  - Se inicializa a cero su contador de edad.
- ❑ Tras cada generación
  - Se incrementa la edad de cada individuo.
  - Se eliminan los que han completado su duración.
- ❑ Controlando la edad se controla la presión selectiva, haciendo innecesaria la selección
- ❑ La introducción de la edad hace variable el tamaño de la población, proporcionando un mecanismo de autorregulación de este tamaño.

# Algoritmos Genéticos con edades

- ❑ En los AG con edades no hay selección.
- ❑ El reemplazo elimina los individuos que han sobrepasado su duración.
- ❑ El tamaño de la descendencia depende del tamaño actual de la población:

$$tam\_descendencia[t] := \lfloor \rho \cdot tam\_pob[t] \rfloor$$

- ❑ Donde  $\rho$  es el coeficiente de reproducción (valores próximos a 0,4).
- ❑ La calidad del AG mejora al aumentar  $\rho$  a costa de incrementar el coste computacional.

# Algoritmos Genéticos con edades

```
funcion AGConEdad(){
P[0] = Poblacion_inicial(); //asignando edad
AptP[0] = Evaluacion(P[0]);
EdadesP[0] = Edad_poblacion(P[0]);
t = 0;
mientras (t <Num_max_gen) y no CondTermina(P[t], AptP[t]){
    EdadesP[t]++;
    Q[t] = Reproduccion(P[t], ρ); //al azar
    Q[t] = Mutacion(Q[t]);
    P[t] = Mezclar(P[t], Q[t]);
    AptP[t] = Evaluacion(P[t]);
    EdadesP[t] = Nuevas_edades(P[t]);
    P[t] = EliminarAntiguos(P[t], AptP[t], EdadesP[t]);
    t++;
}
```

# Algoritmos Genéticos con edades

- Como no hay selección todos los individuos de la población pueden ser elegidos como progenitores de forma equiprobable.
- A cada individuo se le asigna su duración al ser evaluado por primera vez (valor que permanece constante hasta que el individuo desaparece por superar su duración).
- Tamaño de la población en la t-ésima generación:

$$Tam\_pob[t+1] = Tam\_pob[t] + tam\_descendencia[t] - Eliminados[t]$$

- Donde *Eliminados[t]* es el número de individuos que han rebasado su duración en la iteración

# Algoritmos Genéticos con edades

- ❑ Requisitos del criterio de asignación de duraciones:
  - Reforzar la presencia de los más aptos
  - Controlar el tamaño de la población.
- ❑ La descendencia de un individuo es directamente proporcional a su duración: se asignan duraciones más altas a los individuos más aptos.
- ❑ La duración no debe ser estrictamente proporcional a la aptitud para no violar el segundo requisito.
- ❑ Al calcular la duración de un individuo conviene considerar, además de la aptitud, el estado de la búsqueda, dado por las aptitudes media y extremas de la población.
  
- ❑ Veamos 3 formas de asignar edades a los individuos

# Algoritmos Genéticos con edades

- ❑ Asignación proporcional

$$TiempoVida(\mathbf{v}) = \min\{MaxTV, MinTV + DifTV \frac{Aptitud(\mathbf{v})}{AptMed}\}$$

- ❑ Asignación lineal

$$TiempoVida(\mathbf{v}) = MinTV + 2 DifTV \frac{Aptitud(\mathbf{v}) - AptMin}{AptMax - AptMin}$$

Las cotas a las duraciones  $MaxTV$  y  $MinTV$  son parámetros con valores por defecto 7 y 1.

$$DifTV := \frac{1}{2} (MaxTV - MinTV)$$

# Algoritmos Genéticos con edades

## ❑ Asignación bilineal

$$TiempoVida(\mathbf{v}) = \begin{cases} MinTV + DifTV \frac{Aptitud(\mathbf{v}) - AptMin}{AptMed - AptMin} & \text{sii } Aptitud(\mathbf{v}) \leq AptMed \\ \frac{1}{2}(MinTV + MaxTV) + DifTV \frac{Aptitud(\mathbf{v}) - AptMed}{AptMax - AptMed} & \text{sii } Aptitud(\mathbf{v}) > AptMed \end{cases}$$

# Algoritmos Genéticos con edades

- ❑ Con la asignación proporcional las probabilidades de supervivencia son proporcionales a su aptitud relativa.
- ❑ La cota  $MaxTV$  se añade para evitar el crecimiento descontrolado de la población.
- ❑ La asignación lineal se introduce porque la experiencia indica que conviene considerar también la aptitud de los individuos respecto a la mejor aptitud obtenida hasta entonces (aptitud objetiva):

$$AptObj(\mathbf{v}) := \frac{Aptitud(\mathbf{v})}{AptMax(P)}$$

## Algoritmos Genéticos con edades

- ❑ La asignación lineal tiene el inconveniente de que cuando hay muchos individuos con aptitudes similares a la mejor, se asignan largas duraciones a todos ellos (crece significativamente el tamaño de la población).
- ❑ Con la estrategia de asignación bilineal se trata de rebajar esta tendencia.
- ❑ La estrategia de asignación lineal es la que proporciona mayor precisión, pero es la más costosa.
- ❑ La menos costosa es la bilineal, pero no tiene buena precisión.
- ❑ La asignación proporcional representa un punto intermedio.

## Tamaño de población en AG con edades

- ❑ En un AG con edades el tamaño de la población evoluciona de forma oscilante:
  - Al principio, cuando la variedad de aptitudes es relativamente alta, el tamaño de la población crece (búsqueda en anchura del óptimo).
  - Una vez que ha sido localizada la vecindad del óptimo, el algoritmo comienza a converger y el tamaño de la población se reduce.
  - Cuando se consigue una mejoría tiene lugar otra explosión demográfica, seguida de otra etapa de convergencia.

# Algoritmos genéticos diploides

- ❑ Son AGs con representación duplicada (en biología diploides).
- ❑ En cada gen hay lugar para dos alelos.
- ❑ Uno de los alelos es dominante y se expresará al calcular la aptitud del individuo.
- ❑ En la reproducción, cada individuo hereda en cada gen un alelo al azar de cada uno de sus progenitores.
- ❑ La duplicación de los alelos evita la destrucción prematura de información que pudiera resultar valiosa en el futuro.
- ❑ Especialmente útil cuando se trabaja con funciones de aptitud cambiantes en el tiempo.

# Dominancia

- Por ejemplo, si se considera como criterio para cada uno de los locus la dominancia de mayúsculas, se tendrá:
- De los dos genotipos

**AbCdE**

**abCDE**

- El fenotipo tendrá las características **AbCDE**.
  - Las características dominantes se expresan en homocigotos ( $AA \rightarrow A$ ) y heterocigotos ( $Aa \rightarrow A$ ).
  - Las características recesivas solamente se expresan en homocigotos ( $aa \rightarrow a$ ).

# Búsqueda multiobjetivo

- En muchos problemas se necesita optimizar simultáneamente varios objetivos  $f_1(\mathbf{x}), \dots, f_k(\mathbf{x})$
- Matemáticamente:

$$\left\{ \begin{array}{ll} \text{maximizar} & f_k(\mathbf{x}) \quad (\forall k = 1, \dots, r) \\ \text{sujeto a} & \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^m \end{array} \right.$$

- El óptimo global ya no es inmediato:
  - ¿Optimizar una función sacrificando el resto?
  - ¿Calcular la media de todas las funciones?

Modelo	Autonomía (km)	Peso (kg)	Precio
B1	40	33	1000
B2	50	20	1200
B3	50	20	1100

Ejemplo: Elegir bici. Problema de 3 objetivos donde quiero maximizar la autonomía a la vez que minimizar el peso de la bicicleta y su precio

## Búsqueda multiobjetivo

- ❑ A veces los objetivos  $f_k(x)$ , admiten una base común de medidas (por ejemplo, costes o beneficios equivalentes).
  - El problema puede reducirse a uno monoobjetivo mediante una **función combinada de objetivos**:

$$f(\mathbf{x}) := w_1 f_1(\mathbf{x}) + \cdots + w_r f_r(\mathbf{x})$$

- $w_i$  Son los factores de ponderación, que por convenio suman 1.
- La elección de los  $w_k$  es fundamental

# Búsqueda multiobjetivo

- En muchos problemas se necesita optimizar simultáneamente varios objetivos  $f_1(\mathbf{x}), \dots, f_r(\mathbf{x})$

$$\begin{cases} \text{maximizar} & f_k(\mathbf{x}) \quad (\forall k = 1, \dots, r) \\ \text{sujeto a} & \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^m \end{cases}$$

- Soluciones basadas en Óptimo de Pareto:***

- Una solución  $\mathbf{x} \in \mathcal{X}$  es un **óptimo de Pareto** cuando **no** existe ninguna solución mejor en ningún objetivo: soluciones no dominadas

# Búsqueda multiobjetivo

- Soluciones no dominadas u óptimos de Pareto:
  - Una solución  $x \in \mathcal{X}$  es un óptimo de Pareto cuando no existe ninguna solución mejor en ningún objetivo:

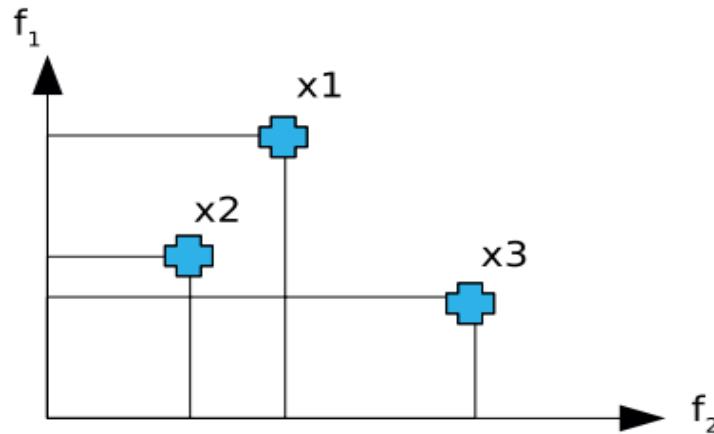
$$\nexists y \in \mathcal{X}$$

tal que

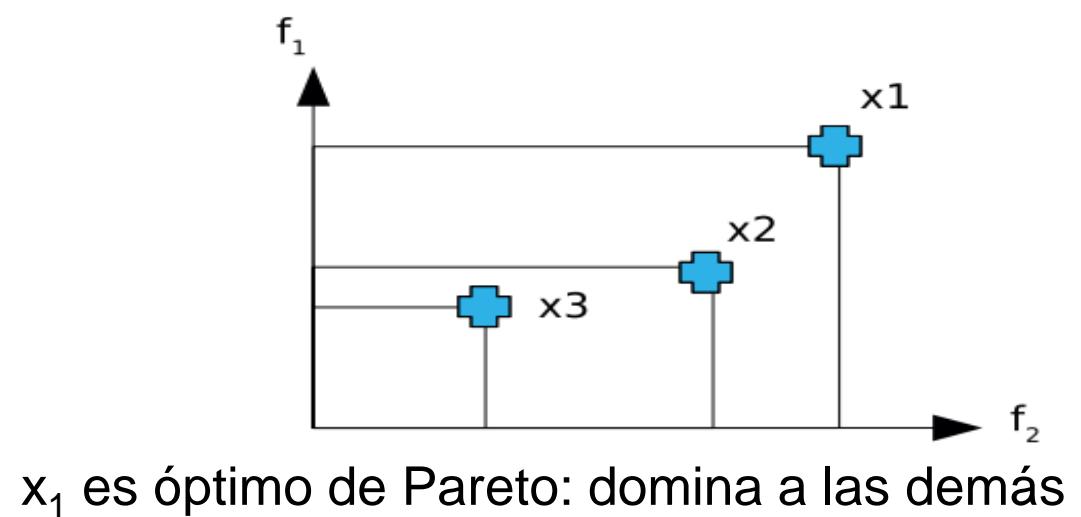
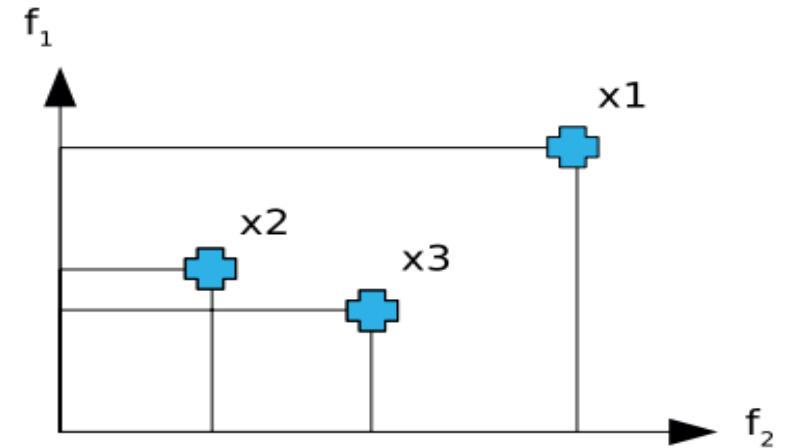
- $f_i(y) \geq f_i(x) (\forall i = 1, \dots, k)$  y
- $f_j(y) > f_j(x)$  para algún  $j$ .

- Se espera que las mejores soluciones estén entre ellas.

# Búsqueda multiobjetivo



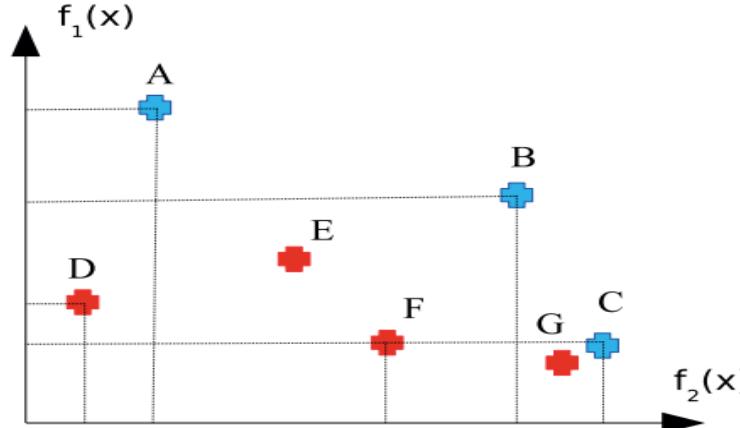
$x_1$  no es óptimo de Pareto: solo domina a  $x_2$



$x_1$  es óptimo de Pareto: domina a las demás

## Búsqueda multiobjetivo

- Las soluciones válidas forman la frontera de Pareto (A,B y C) y el resto son soluciones dominadas:



- Enfoques en el Algoritmo evolutivo:
  - Eliminar los (puntos) cromosomas dominados
  - Reproducir con mayor probabilidad los que están en la frontera. Así, la frontera irá expandiéndose conforme pasan las generaciones
  - Utilizar en la selección mediante un ranking según el número de soluciones que domina cada cromosoma, etc.

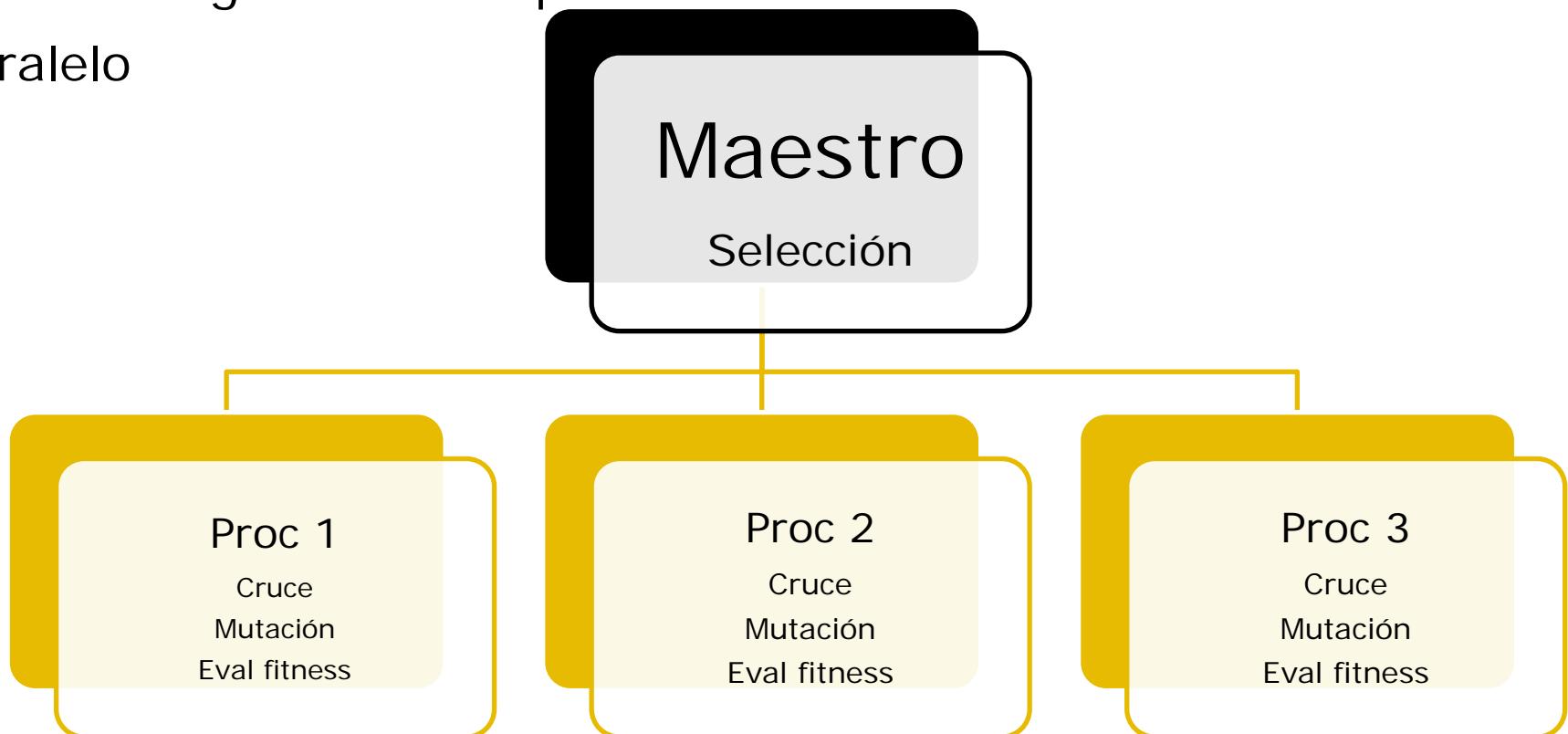
# Algoritmos evolutivos Multiobjetivo

- ❑ Algoritmos que no utilizan el concepto de dominancia
  - Funciones agregativas lineales
$$f(x) = w_1 f_1(x) + w_2 f_2(x) + \cdots + w_k f_k(x)$$
- ❑ Algoritmos que sí utilizan el concepto de dominancia
  - VEGA: selección proporcional según las funciones a optimizar
  - MOGA (*Multi-Objective Genetic Algorithm*)
  - **NSGA-II** (*Elitist Non-Dominated Sorting Genetic Algorithm*)
  - SPEA, SPEA2, MOMGA, MOMGA-II, PAES. . .
- ❑ Aplicaciones: Planificación, empaquetado, diseño de circuitos, diseño de sistemas de control, detección de ataques en redes, procesamiento de imágenes, problemas de predicción, optimización de parámetros en mercados financieros\*, etc.

# Algoritmos Evolutivos Paralelos

## □ Paralelización global

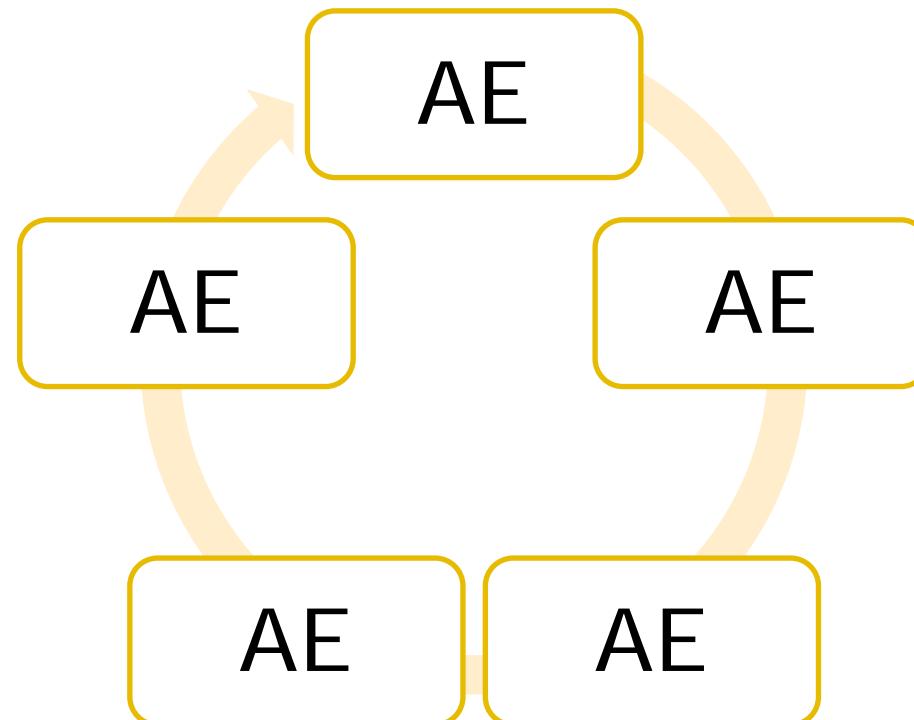
- Misma estructura del AG: población única
- Aplicación de operadores genéticos en paralelo
- Evaluación en paralelo



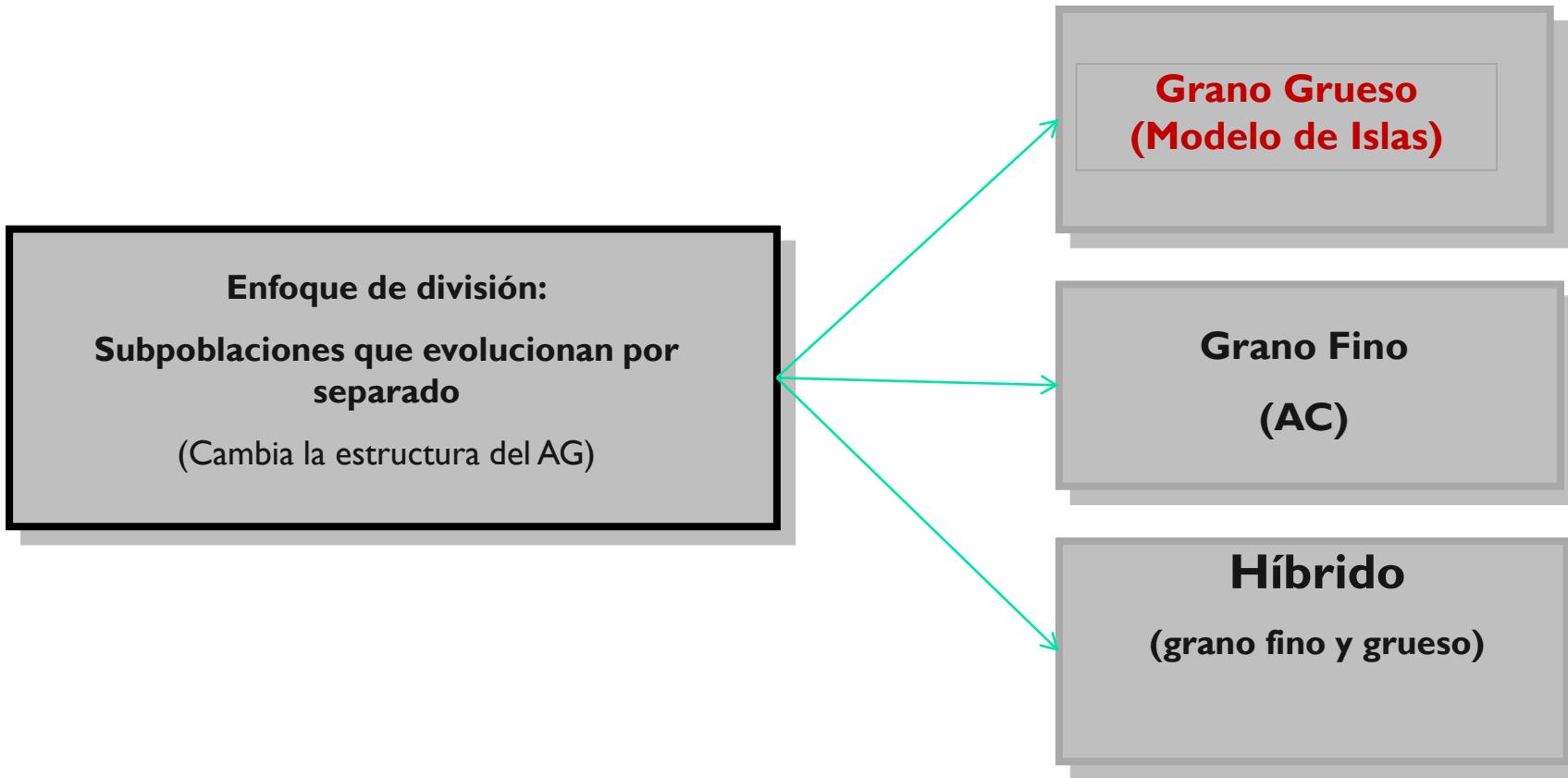
# Algoritmos Evolutivos Paralelos

## □ Enfoque de división en subpoblaciones

- La población se divide en subpoblaciones o individuos
- Cada subpoblación o individuo en un procesador

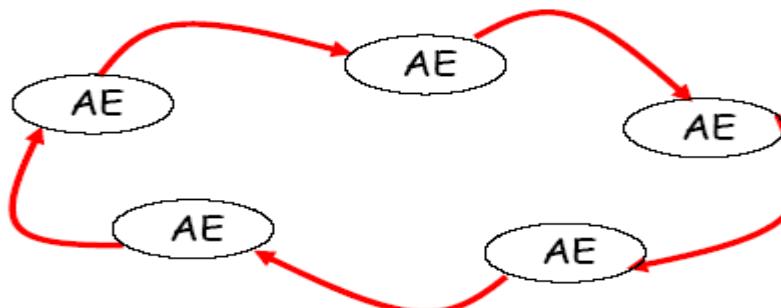


# Algoritmos Evolutivos Paralelos



## Modelo de islas (distribuido)

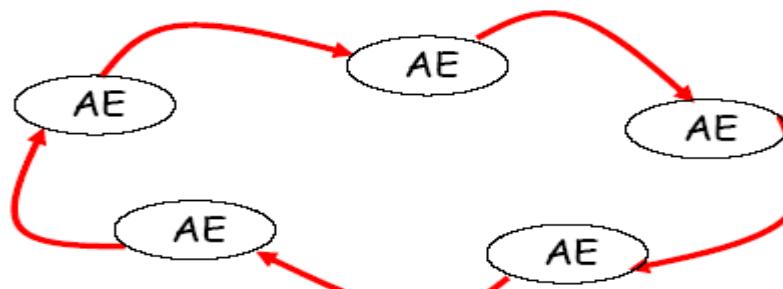
- ❑ Se divide la población en **subpoblaciones** mas pequeñas que evolucionan independientemente en cada procesador o isla.
- ❑ Cada **subpoblación** evoluciona independientemente en cada procesador o isla.
- ❑ En cada procesador se lleva a cabo un Algoritmo Evolutivo (AE)
- ❑ En cada isla la población inicial se genera con diferente semilla.
- ❑ Cada cierto número de generaciones (*epoch*) intercambian individuos entre las poblaciones: migraciones. Esto genera diversidad genética.
- ❑ Podemos emplear distintos operadores en cada isla



## Modelo de islas (distribuido)

- ❑ Parámetros adicionales:
  - Número de generaciones entre intercambios.
  - Número de individuos a intercambiar.
  - Selección de los individuos a enviar.
  - Selección de los individuos a reemplazar por los recibidos.

La implementación del envío y recepción de individuos es sencillo en un entorno de paso de mensajes



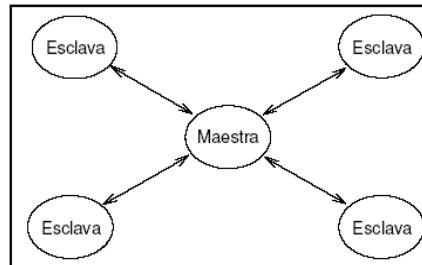
# Modelo de islas (distribuido)

- ❑ ¿Cada cuánto tiempo debemos intercambiar individuos?
  - Demasiado rápido y todas las poblaciones convergen hacia la misma solución.
  - Demasiado lento y tomará mucho tiempo lograr una solución
  - Número habitual de generaciones por epoch: entre 25 y 150
- ❑ ¿Cuántos y cuáles individuos intercambiar?
  - Depende del tamaño de la población (normalmente entre 2 y 5)
  - Normalmente, un mayor número de poblaciones proporciona mejores resultados
  - Es mejor migrar individuos seleccionados al azar que siempre los mejores
  - Para el reemplazo podemos seleccionar los individuos aleatoriamente o tomar los peores

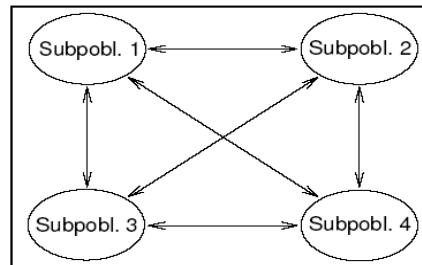
# Modelo de islas (distribuido)

- Se pueden distinguir diferentes modelos de islas en función de la comunicación entre las subpoblaciones.

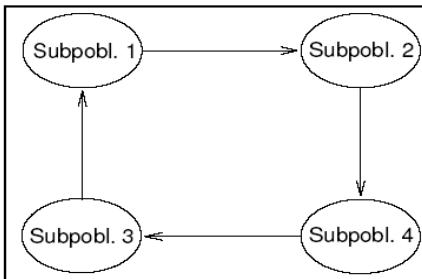
- Estrella



- En red

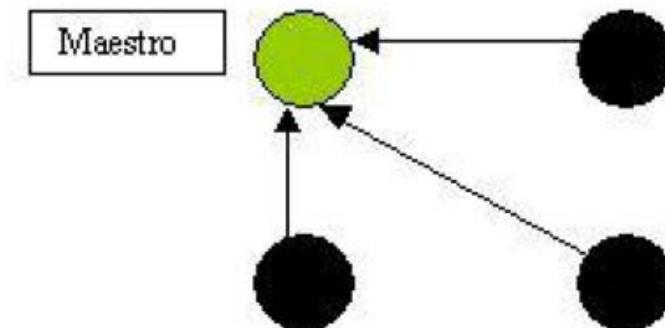
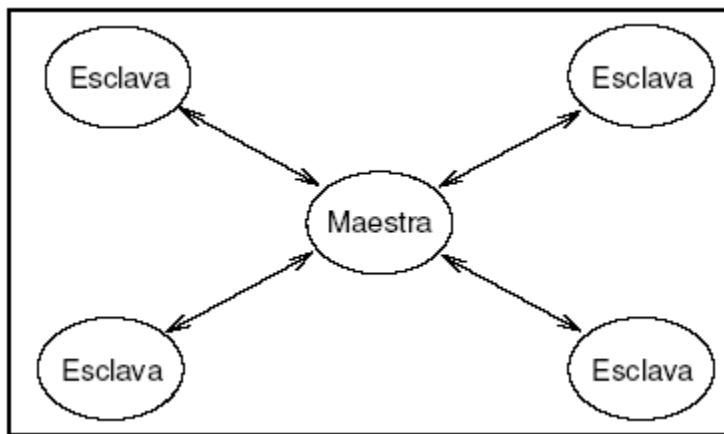


- En anillo



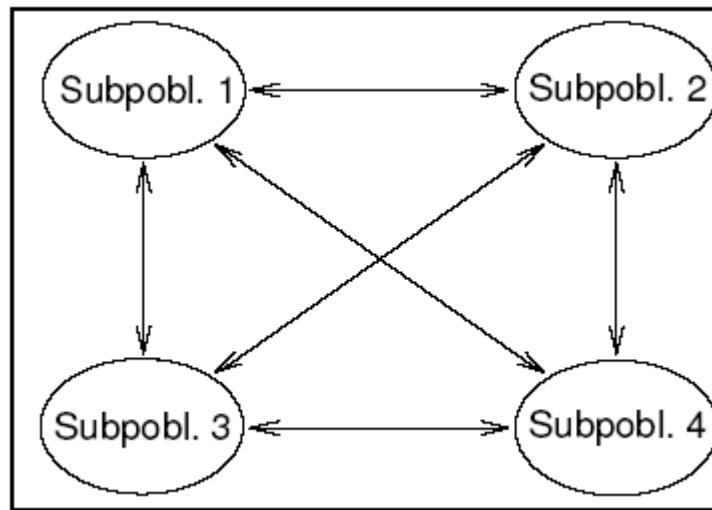
## Modelo en estrella: maestro-esclavo

- Una subpoblación se selecciona como maestra (la que tiene mejor media en el valor de la función objetivo), siendo las demás consideradas como esclavas.
- Todas las subpoblaciones esclavas mandan sus  $h_1$  mejores individuos ( $h_1 > 1$ ) a la subpoblación maestra, la cual a su vez manda sus  $h_2$  mejores individuos ( $h_2 > 1$ ) a cada una de las subpoblaciones esclavas.



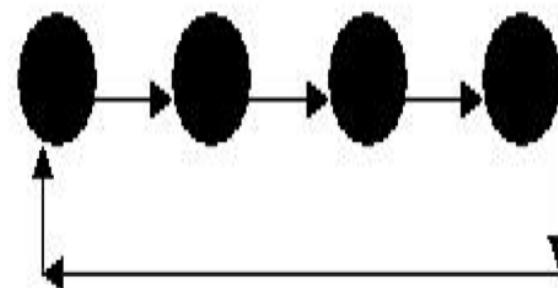
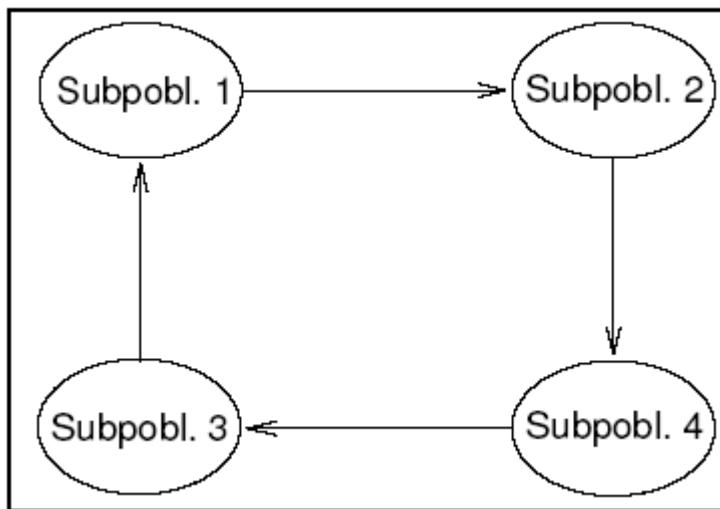
## Modelo en red: todos con todos

- No hay una jerarquía entre las subpoblaciones; todas mandan sus  **$h3$**  ( $h3>1$ ) mejores individuos al resto de las subpoblaciones.



## Modelo en anillo

- Cada subpoblación envía sus  $h4$  mejores individuos ( $h4 > 1$ ), a una población vecina, efectuándose la migración en un único sentido de flujo.

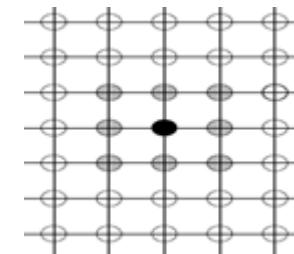


# Algoritmo

```
set num_Gen to 0;  
  
initialize Population(num_Gen);  
  
evaluate fitness of Population(num_Gen);  
  
while (not termination_condition) {  
  
    num_Gen++;  
  
    select Parents(num_Gen) from Population(num_Gen - 1);  
    apply crossover to Parents(num_Gen) to produce Offspring(num_Gen);  
    apply mutation to Offspring(num_Gen) to get Population(num_Gen);  
    apply migration to Population(num_Gen);  
    evaluate Population(num_Gen);  
}
```

## Grano fino: modelo celular

- ❑ Simula relaciones personales entre individuos de una misma localidad.
  - Los individuos se disponen en una parrilla de dos dimensiones con un individuo en cada una de las posiciones de la rejilla.
  - Cada individuo un procesador
  - La evaluación se realiza simultáneamente para todos los individuos
  - La selección, reproducción y cruce se hace de forma local con un reducido número de vecinos.
  - Con el tiempo se van formando grupos de individuos que son homogéneos genéticamente como resultado de la lenta difusión de individuos.
  - Parámetros a tener en cuenta
    - Tamaño de la vecindad
    - Topología de los procesadores
    - Esquema de reemplazamiento de individuos
    - ...



# Swarm Intelligence (Inteligencia Colectiva)

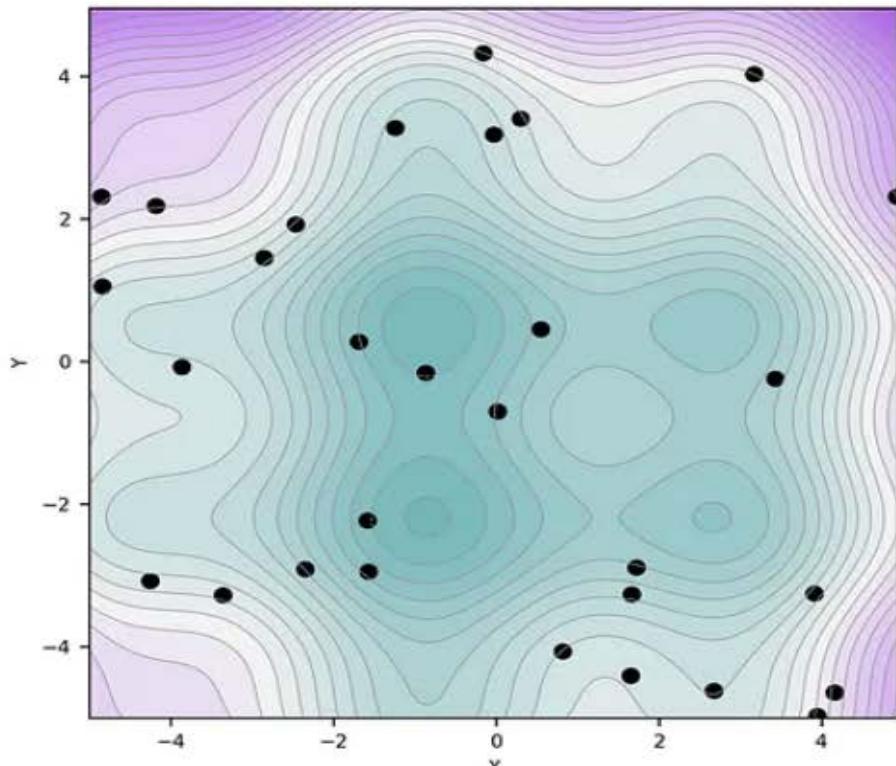
- ❑ Se basa en el comportamiento colectivo de sistemas naturales o artificiales descentralizados y auto organizados
- ❑ Suelen utilizar una población de agentes muy simples o *boids* que interactúan entre sí y con su entorno.
- ❑ Los agentes tiene una reglas muy simples y las interacciones entre los agentes hacen que surja un comportamiento global “inteligente”
- ❑ Ejemplos: Colonias de hormigas, termitas, abejas, bandadas de pájaros, bancos de peces...

# PSO: Optimización basada en grupos de partículas

## PSO: Particle Swarm Optimization

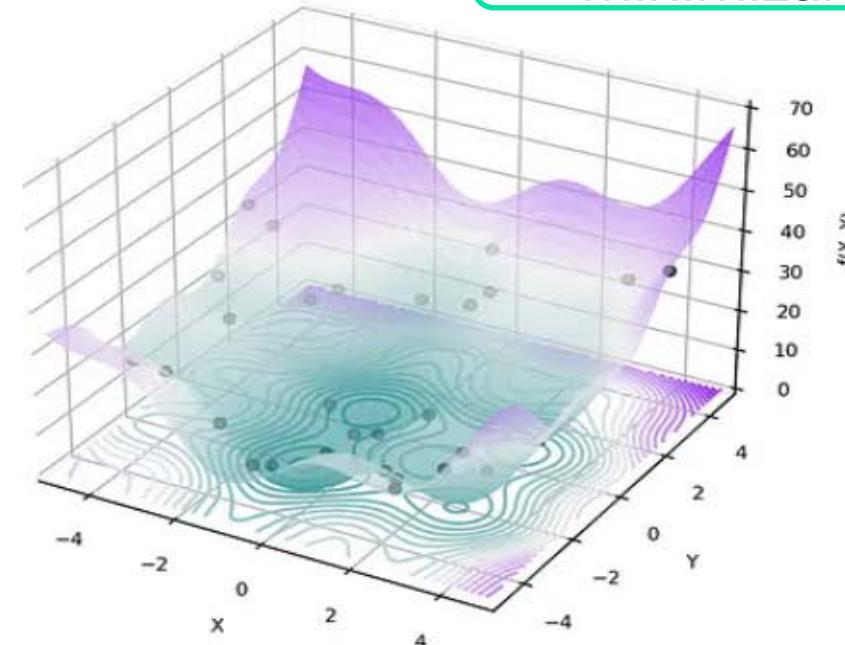
- Inspirado en los patrones de vuelo de las aves
- Simula los movimientos de una población de aves buscando comida: la estrategia es seguir al que está más cerca de la comida
- Una población de agentes (partículas), cuyas posiciones en un espacio multidimensional representan posibles soluciones, se mueven actualizando la velocidad según la información recogida por el grupo (llamado enjambre).
- En cada iteración, cada partícula está guiada por su mejor posición anterior y por la mejor posición global.

# PSO: Particle Swarm Optimization



$$f(x, y) = x^2 + (y + 1)^2 - 5\cos(1.5x + 1.5) - 3\cos(2x - 1.5)$$

Random Initialization of particles position [Original Image]

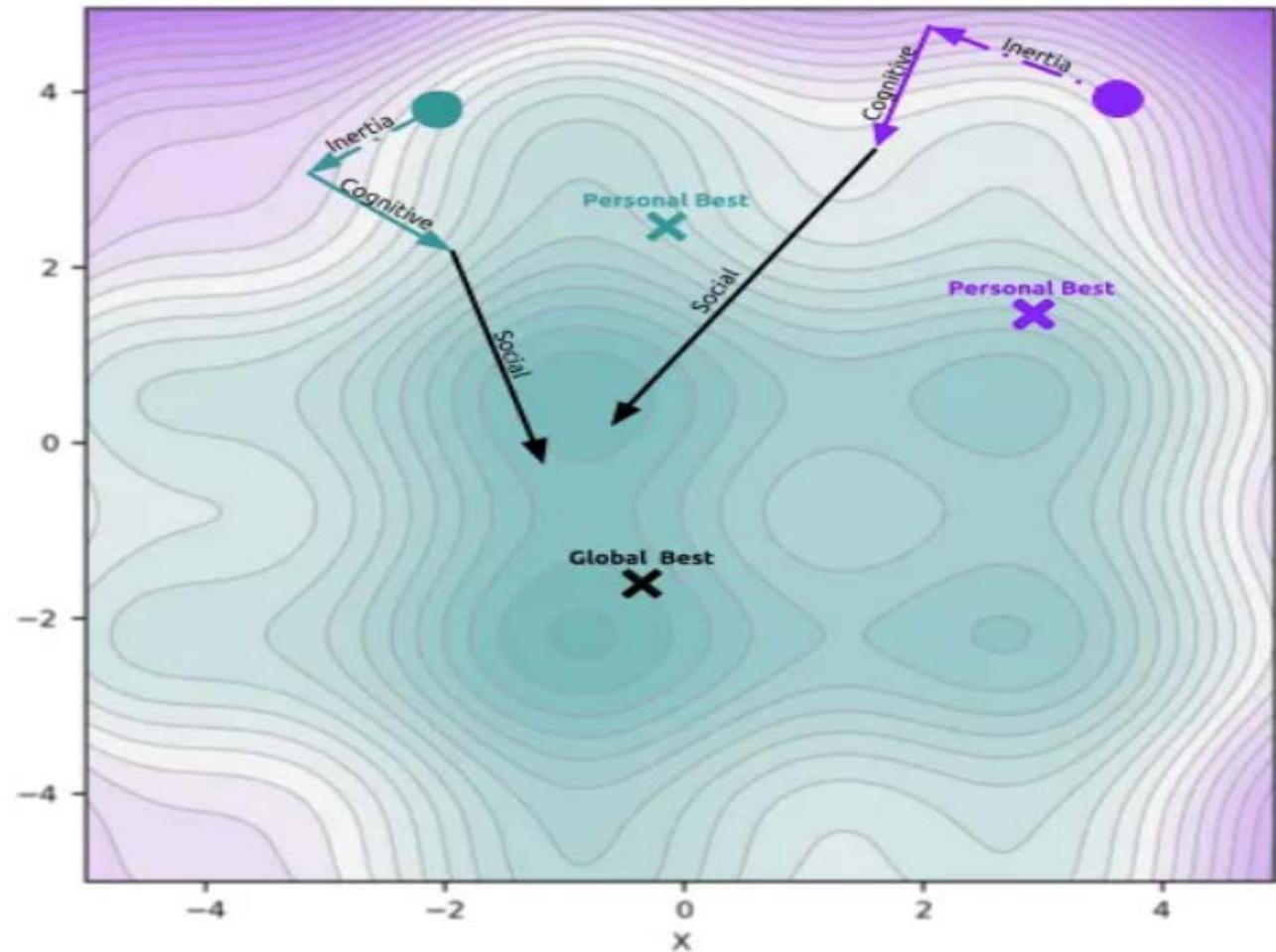


Función a  
minimizar

Axel Thevenot  
Towards Data Science

# PSO: Particle Swarm Optimization

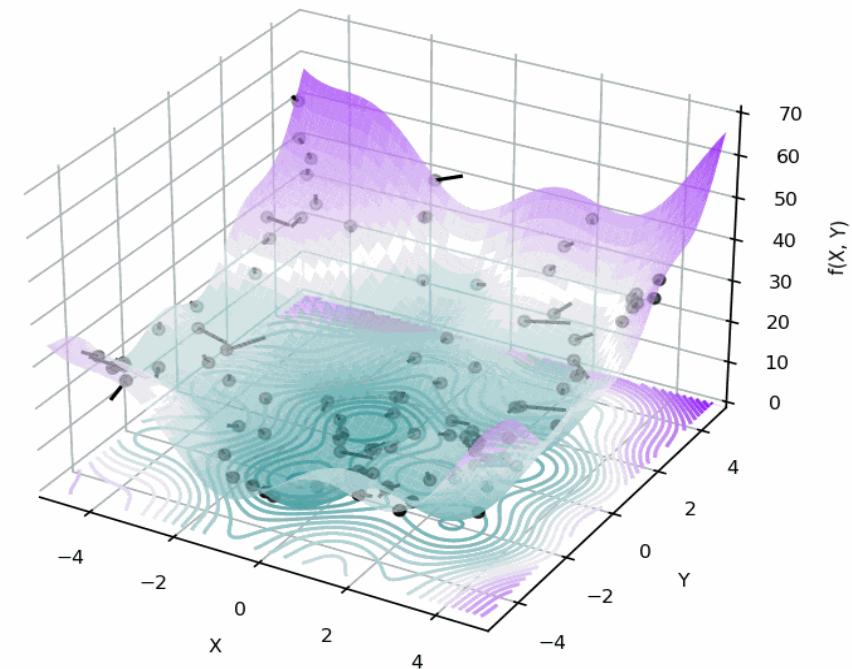
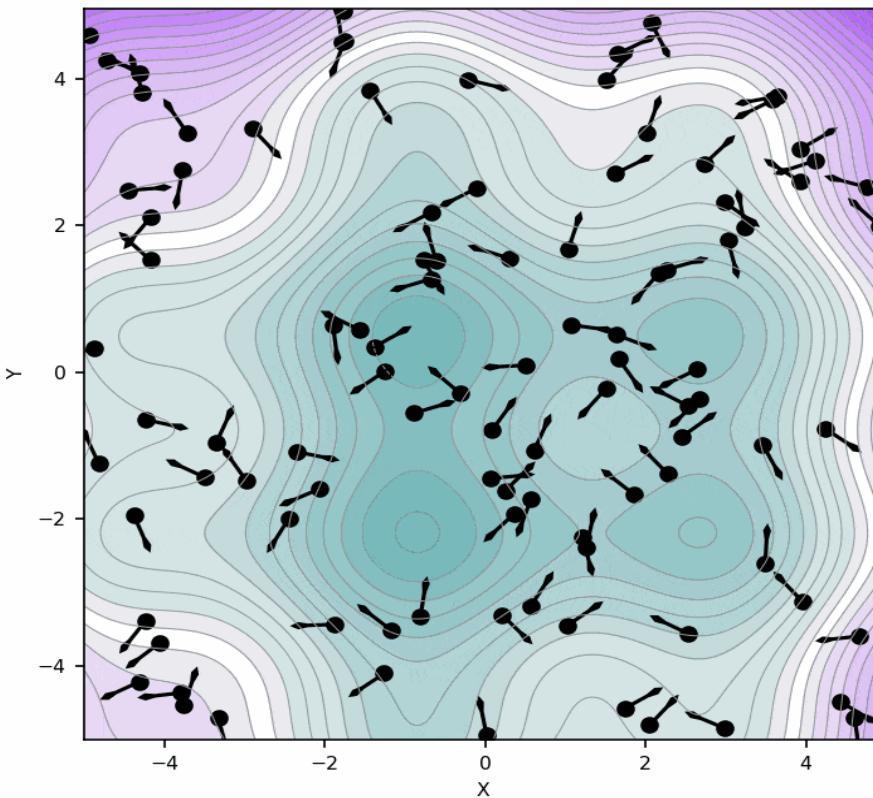
En cada iteración en el espacio de búsqueda, la velocidad de cada partícula se acelera hacia su mejor posición anterior (mejor marca personal) y hacia la mejor solución del grupo (mejor global).



Axel Thevenot  
Towards Data Science

# PSO: Particle Swarm Optimization

[1/100]  $w:0.800 - c_1:2.000 - c_2:2.000$



Axel Thevenot  
Towards Data Science

# Algoritmo PSO

Inicializa un grupo de partículas al azar (soluciones). En cada iteración, cada partícula se actualiza según dos "mejores" valores.

- $pbest$  : la mejor solución (fitness) lograda hasta ahora por esa partícula.
- $gbest$  : el mejor valor obtenido hasta el momento por cualquier partícula de la población. Mejor global
- Cada partícula actualiza su velocidad y posición

Do

Para cada partícula

Calcular el mejor fitness obtenido hasta ahora **pBest**

Elegir la partícula con mejor fitness de todo el grupo **gBest**

Para cada partícula

Calcular velocidad según ecuación

$v[] = v[] + c1 * \text{rand}() * (\text{pBest}[] - pos[]) + c2 * \text{rand}() * (\text{gBest}[] - pos[])$

Actualizar la posición según ecuación

$pos[] = pos[] + v[]$

While no fin (iteraciones o error)

$$P_i^{t+1} = P_i^t + V_i^{t+1}$$

$$V_i^{t+1} = wV_i^t + c_1 r_1 (P_{best(i)}^t - P_i^t) + c_2 r_2 (P_{bestglobal}^t - P_i^t)$$

Inertia

Cognitive (Personal)

Social (Global)

Particle update [Original Image]

# Algoritmo PSO

```
public void run () {
    Particle[] particles = initialize();
    . . .
    for (int i = 0; i < epochs; i++) {
        if (bestEval < oldEval) {
            oldEval = bestEval;
        }

        for (Particle p : particles) {
            p.updatePersonalBest();
            updateGlobalBest(p);
        }

        for (Particle p : particles) {
            updateVelocity(p);
        }
    }
}
```

# Algoritmo PSO

```

private void updateVelocity (Particle particle) {
    Vector oldVelocity = particle.getVelocity();
    Vector pBest = particle.getBestPosition();
    Vector gBest = bestPosition.clone();
    Vector pos = particle.getPosition();

    .
    .

    double r1 = random.nextDouble();
    double r2 = random.nextDouble();

    // The first product of the formula.
    Vector newVelocity = oldVelocity.clone();
    newVelocity.mul(inertia);
}

```

$$P_i^{t+1} = P_i^t + V_i^{t+1}$$

$$V_i^{t+1} = wV_i^t + c_1r_1(P_{best(i)}^t - P_i^t) + c_2r_2(P_{bestglobal}^t - P_i^t)$$

Inertia

Cognitive (Personal)

Social (Global)

Particle update [Original Image]

# Algoritmo PSO

```
// The second product of the formula.
pBest.sub(pos);
pBest.mul(cognitiveComponent);
pBest.mul(r1);
newVelocity.add(pBest);
```

```
// The third product of the formula.
gBest.sub(pos);
gBest.mul(socialComponent);
gBest.mul(r2);
newVelocity.add(gBest);
```

```
particle.setVelocity(newVelocity);
}
```

$$P_i^{t+1} = P_i^t + V_i^{t+1}$$

$$V_i^{t+1} = wV_i^t + c_1r_1(P_{best(i)}^t - P_i^t) + c_2r_2(P_{bestglobal}^t - P_i^t)$$

Inertia
Cognitive (Personal)
Social (Global)

Particle update [Original Image]

# Algoritmo PSO

```
/**  
 * Update the position of a particle by adding its velocity to its position.  
 */  
  
void updatePosition () {  
    this.position.add(velocity);  
}
```

$$P_i^{t+1} = P_i^t + V_i^{t+1}$$

$$V_i^{t+1} = wV_i^t + c_1r_1(P_{best(i)}^t - P_i^t) + c_2r_2(P_{bestglobal}^t - P_i^t)$$

Inertia

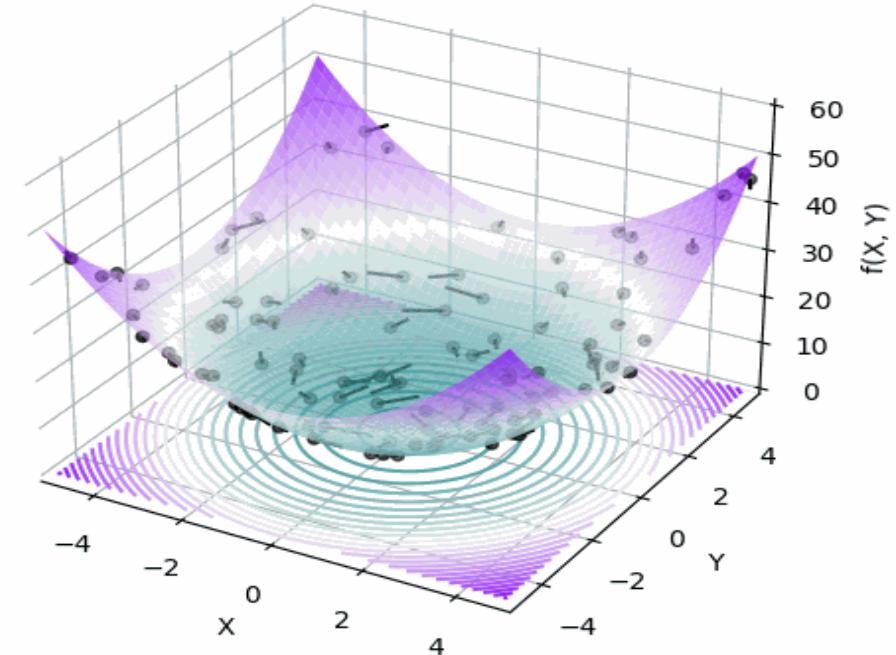
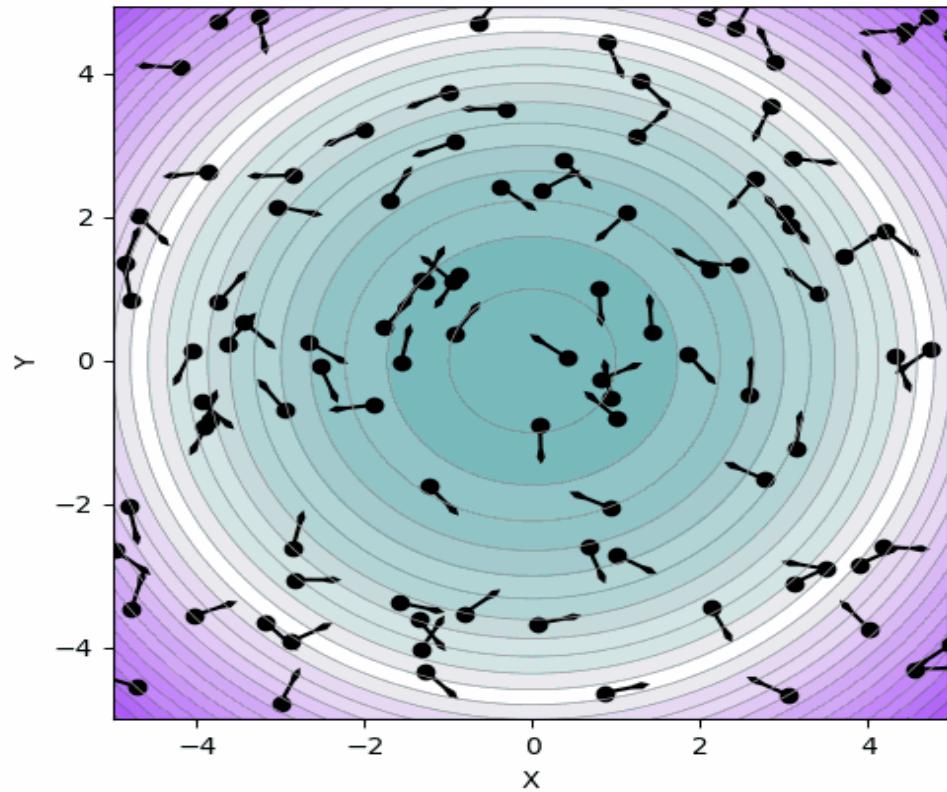
Cognitive (Personal)

Social (Global)

Particle update [Original Image]

# Algoritmo PSO

Sphere function - [1/100] w:0.800 -  $c_1$ :3.500 -  $c_2$ :0.500

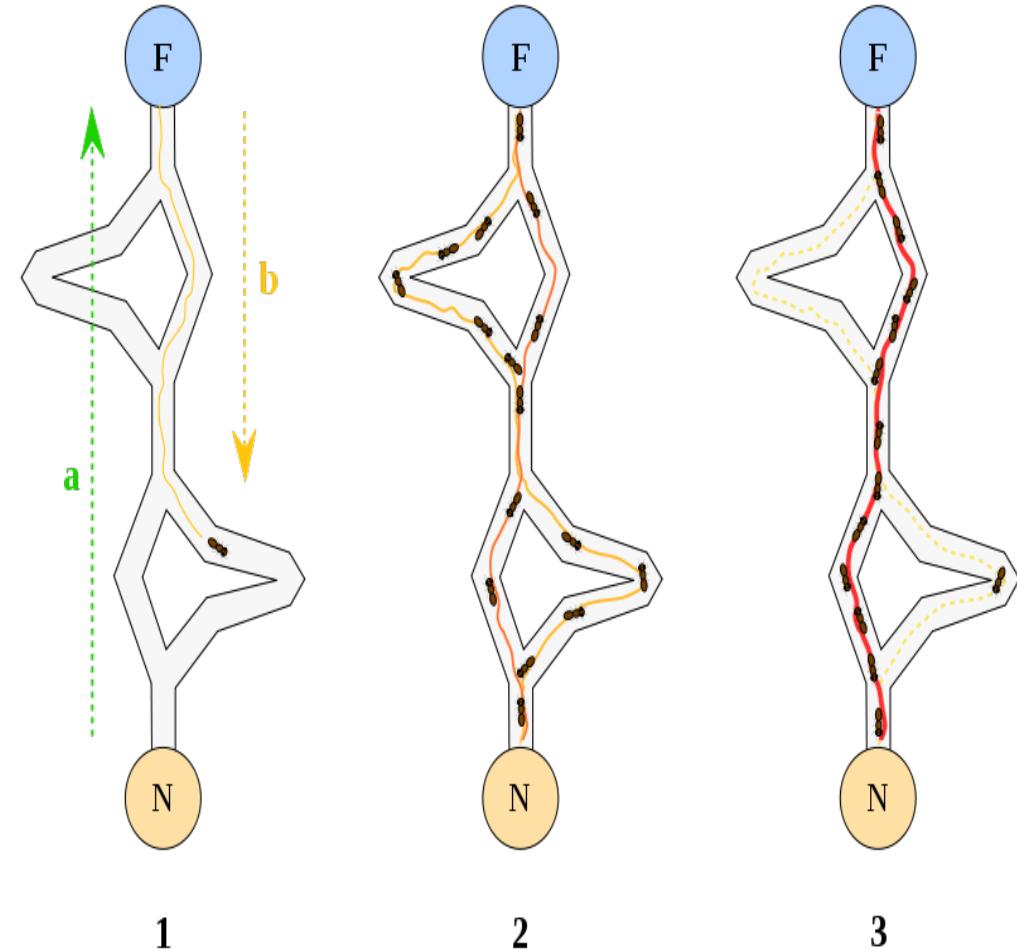


# Optimización basada en colonias de hormigas: ACO

## ACO: Ant Colony Optimization

- ❑ Una hormiga se mueve al azar
- ❑ Si descubre comida la hormiga vuelve al nido dejando a su paso un rastro de feromona atractivo para otras que tenderán a seguir esa pista.
- ❑ Las hormigas refuerzan la ruta más corta pues la visitan más hormigas y el rastro de feromona es mayor.
- ❑ La ruta larga acaba desapareciendo al volatilizarse la feromona. Al final todas las hormigas "eligen" el camino más corto

[http://en.wikipedia.org/wiki/Ant\\_colony\\_optimization](http://en.wikipedia.org/wiki/Ant_colony_optimization).



# Optimización basada en colonias de hormigas: ACO

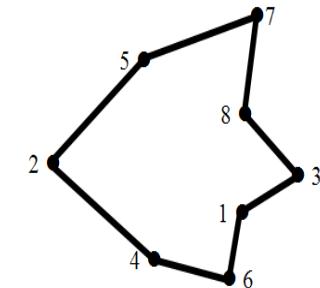
- Aplicación al problema del TSP



## ACO para el TSP

- En cada iteración se "lanza" una colonia de  $m$  hormigas y cada hormiga construye una solución al problema.
- La regla probabilística para el caso del TSP es:

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}(t)]^\alpha \cdot [\eta_{lj}]^\beta} \quad \text{con } j \in N_i^k$$



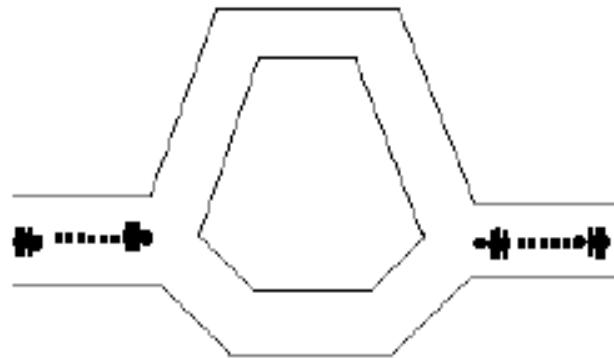
- Probabilidad con la que, en una iteración  $t$  del algoritmo, la hormiga  $k$ , situada actualmente en la ciudad  $i$ , elige a la ciudad  $j$  como próxima parada.
- Conjunto de ciudades no visitadas todavía por la hormiga  $k$ .
- Cantidad de feromona acumulada sobre el arco  $(i,j)$  de la red en la iteración  $t$  (se actualiza en cada iteración y para cada arco).
- Información heurística para la que, en el caso del TSP, se utiliza la inversa de la distancia existente entre las ciudades  $i$  y  $j$ .
- $\alpha$  y  $\beta$  son dos parámetros del algoritmo, los cuales hay que ajustar.

# ACO: Ant Colony Optimization

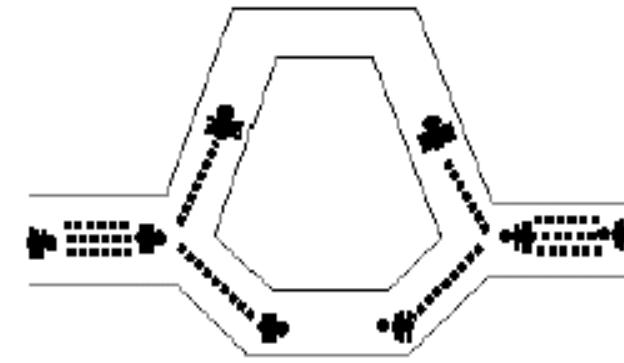
- ❑ Los modelos basados en colonias de hormigas se basan en el comportamiento colectivo de las hormigas en la búsqueda de alimentos para su subsistencia.
- ❑ Insectos casi ciegos, moviéndose aproximadamente al azar, pueden encontrar el camino más corto desde su nido hasta la fuente de alimentos y regresar.
- ❑ Cuando una hormiga se mueve, deja una señal odorífera, depositando una substancia denominada feromona, para que las demás puedan seguirla.
- ❑ En principio, una hormiga aislada se mueve esencialmente al azar, pero las siguientes deciden con una buena probabilidad seguir el camino con mayor cantidad de feromonas.

# ACO: Ant Colony Optimization

- En las siguientes figuras se observa como las hormigas establecen el camino más corto.
- En la figura (a) las hormigas llegan al punto en que tienen que decidir por uno de los caminos que se les presenta.



(a)

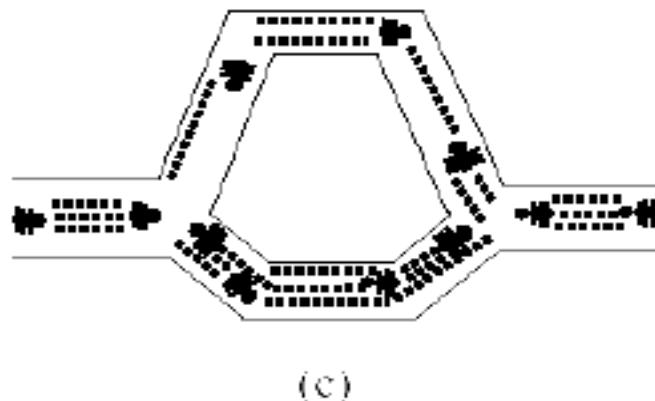


(b)

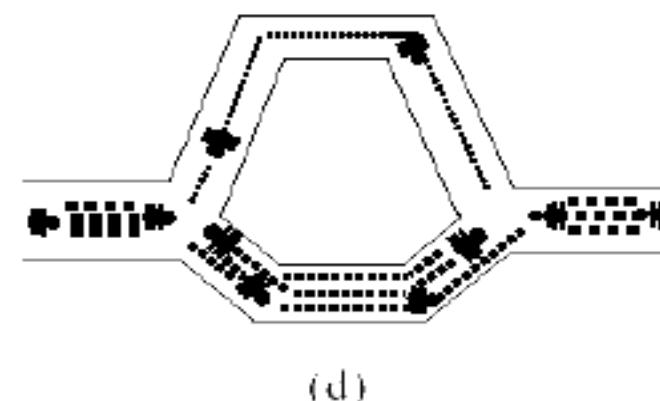
- en (b) realizan la elección de manera aleatoria, algunas hormigas eligen el camino hacia arriba y otras hacia abajo

# ACO: Ant Colony Optimization

- En (c) como las hormigas se mueven aproximadamente a una velocidad constante, las que eligieron el camino más corto alcanzarán el otro extremo más rápido que las otras que tomaron el camino más largo, depositando mayor cantidad de feromona por unidad de longitud;
- En (d) la cantidad de feromona depositada en el trayecto más corto hace que la mayoría de las hormigas elijan este camino, por realimentación positiva, en la que la probabilidad con la que una hormiga escoge un camino aumenta con el número de hormigas que previamente hayan elegido el mismo camino.



(c)



(d)

# ACO: Ant Colony Optimization

- ❑ Se ha utilizado para el problema del viajante (Travelling Salesman Problem TSP).
- ❑ Los algoritmos ACO son procesos iterativos. En cada iteración se "lanza" una colonia de  $m$  hormigas y cada una de las hormigas de la colonia construye una solución al problema.
- ❑ Las hormigas construyen las soluciones de manera probabilística, guiándose por un rastro de feromonas artificiales y por una información calculada a priori de manera heurística. La regla probabilística para el caso del TSP es:

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}(t)]^\alpha \cdot [\eta_{lj}]^\beta} \quad \text{con } j \in N_i^k$$

## ACO: Ant Colony Optimization

- ❑ donde  $p_{ij}^k(t)$  es la probabilidad con la que, en una iteración  $t$  del algoritmo, la hormiga  $k$ , situada actualmente en la ciudad  $i$ , elige a la ciudad  $j$  como próxima parada.
- ❑  $N_i^k$  es el conjunto de ciudades no visitadas todavía por la hormiga  $k$ .
- ❑  $\tau_{ij}(t)$  es la cantidad de feromonas acumulada sobre el arco  $(i,j)$  de la red en la iteración  $t$ .
- ❑  $\eta_{ij}$  es la información heurística para la que, en el caso del TSP, se utiliza la inversa de la distancia existente entre las ciudades  $i$  y  $j$ .
- ❑  $\alpha$  y  $\beta$  son dos parámetros del algoritmo, los cuales hay que ajustar.

# ACO: Ant Colony Optimization

- Cuando todas las hormigas han construido una solución debe actualizarse la feromona en cada arco. La fórmula a seguir es:

$$\tau_{ij}(t+1) = (1 - \rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij}^{best} \quad \Delta\tau_{ij}^{best} = \begin{cases} 1/L^{best} & \text{si el arco } (i,j) \text{ pertenece a } T^{best} \\ 0 & \text{en caso contrario} \end{cases}$$

- donde  $\rho$  es el coeficiente de evaporación de la feromona.  $T^{best}$  puede ser la mejor solución encontrada hasta el momento o bien la mejor solución encontrada en la iteración.  $L^{best}$  es la longitud de la solución  $T^{best}$

## ACO: Ant Colony Optimization

- ❑ Se obliga a que el nivel de feromona esté en el rango  $[\tau_{\min}, \tau_{\max}]$
- ❑ Estos límites se imponen con el objetivo de evitar el estancamiento en la búsqueda de soluciones. Toda la feromona se inicializa con  $\tau_{\max}$
- ❑ Tras la actualización de la feromona puede comenzarse una nueva iteración.
- ❑ El resultado final es la mejor solución encontrada a lo largo de todas las iteraciones realizadas.

# CRO Optimización basada en arrecifes de coral

- ❑ El algoritmo CRO se basa en una simulación artificial del proceso de formación y reproducción de un arrecife de coral.
- ❑ Durante el proceso emula diferentes fases de reproducción y de lucha por el espacio en el arrecife, el cual representa un algoritmo eficiente para resolver problemas de optimización.



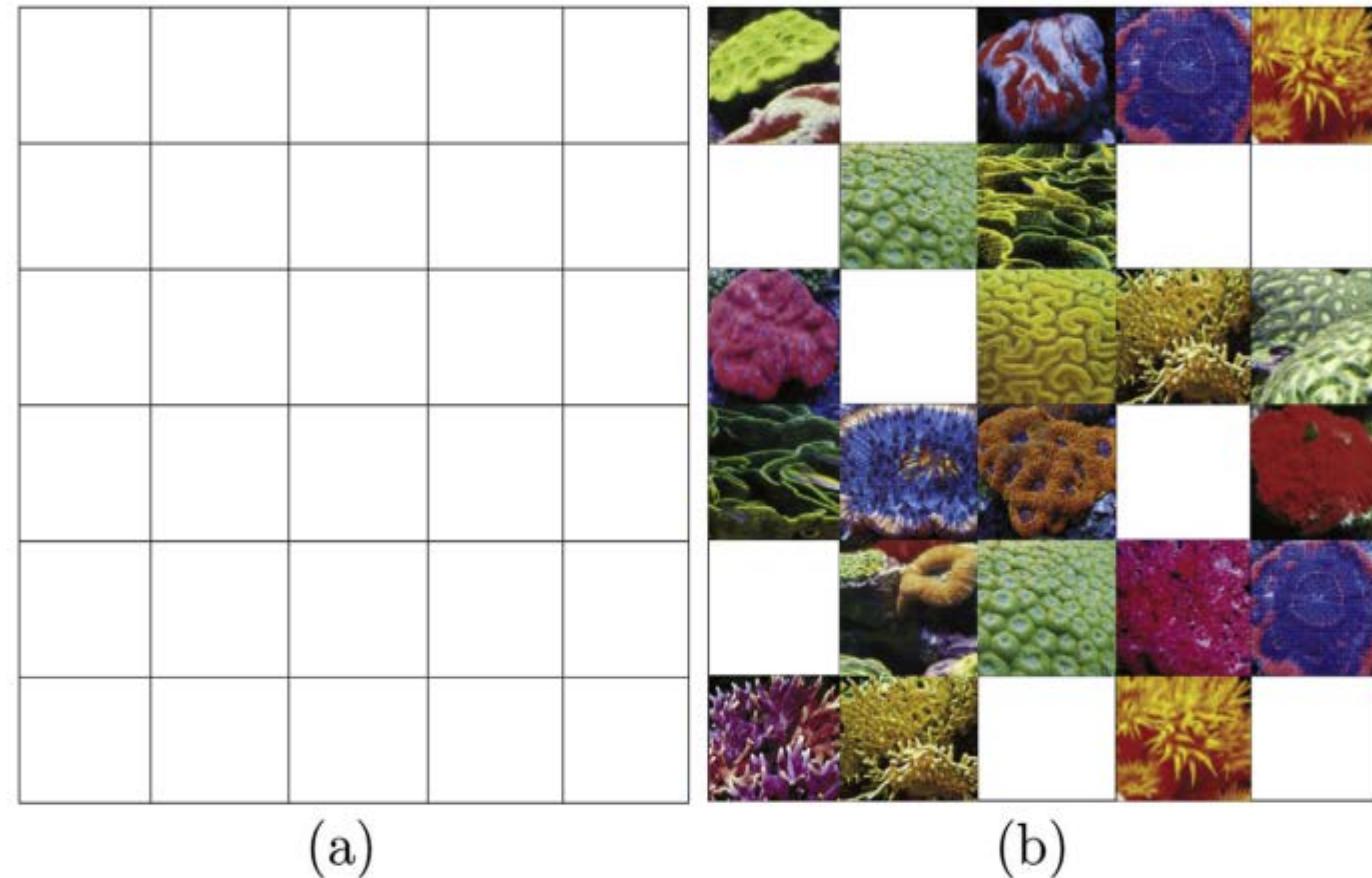
Fuente:

<http://agamenon.tsc.uah.es/Personales/sancho/CRO.html#:~:text=The%20Coral%20Reefs%20Optimization%20CRO,simulated%20to%20solve%20optimization%20problems.>

# CRO Optimización basada en arrecifes de coral

El arrecife **A** es una cuadrícula  $N^*M$ , donde en cada cuadrado  $(i,j)$  se puede alojar un **coral**

Cada coral representa una solución a nuestro problema y se codifica como una cadena de enteros.

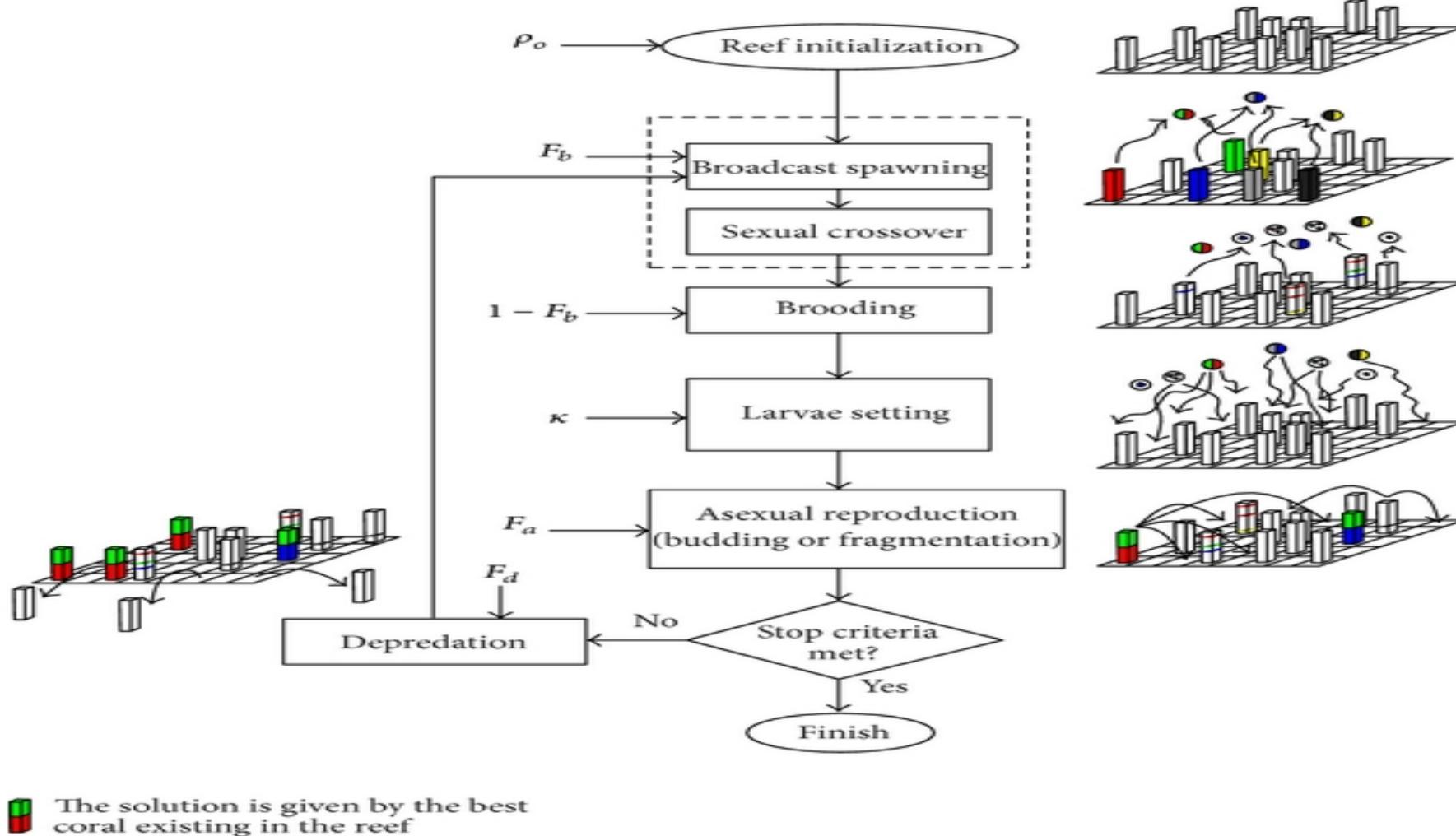


**Fig. 1.** Coral reef simulation; (a) grid; (b) corals and holes in the reef.

## Fuente:

[http://agamenon.tsc.uah.es/Personales/sancho/CRO.html#~:text=The%20Coral%20Reefs%20Optimization%20\(CRO,simulated%20to%20solve%20optimization%20problems.](http://agamenon.tsc.uah.es/Personales/sancho/CRO.html#~:text=The%20Coral%20Reefs%20Optimization%20(CRO,simulated%20to%20solve%20optimization%20problems.)

# CRO Optimización basada en arrecifes de coral

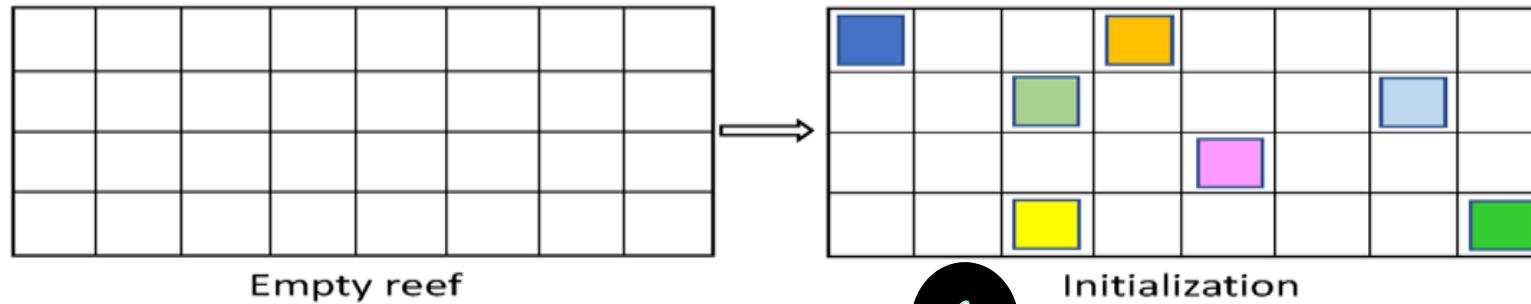


 The solution is given by the best coral existing in the reef

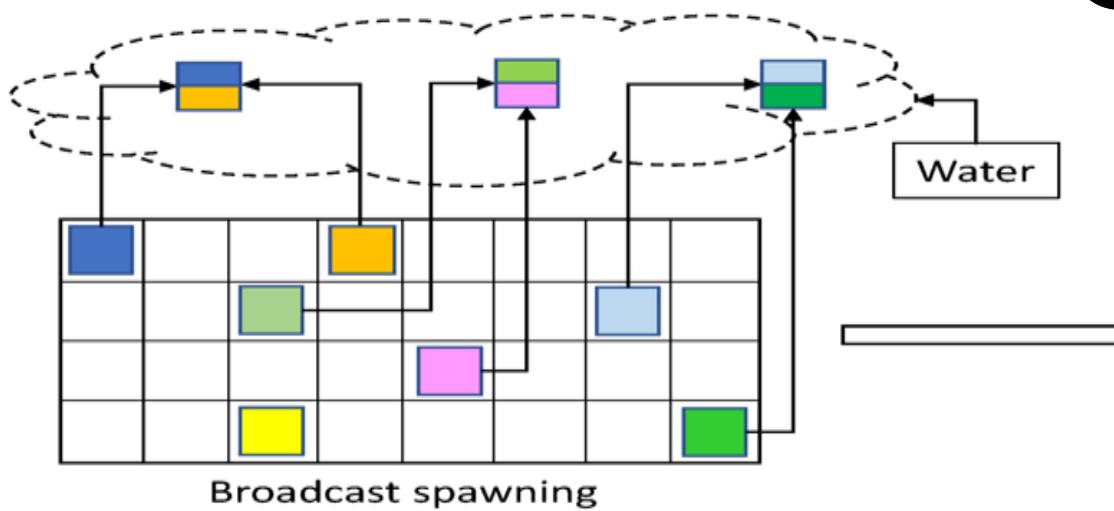
Fuente:

[http://agamenon.tsc.uah.es/Personales/sancho/CRO.html#:~:text=The%20Coral%20Reefs%20Optimization%20\(CRO,simulated%20to%20solve%20optimization%20problems.](http://agamenon.tsc.uah.es/Personales/sancho/CRO.html#:~:text=The%20Coral%20Reefs%20Optimization%20(CRO,simulated%20to%20solve%20optimization%20problems.)

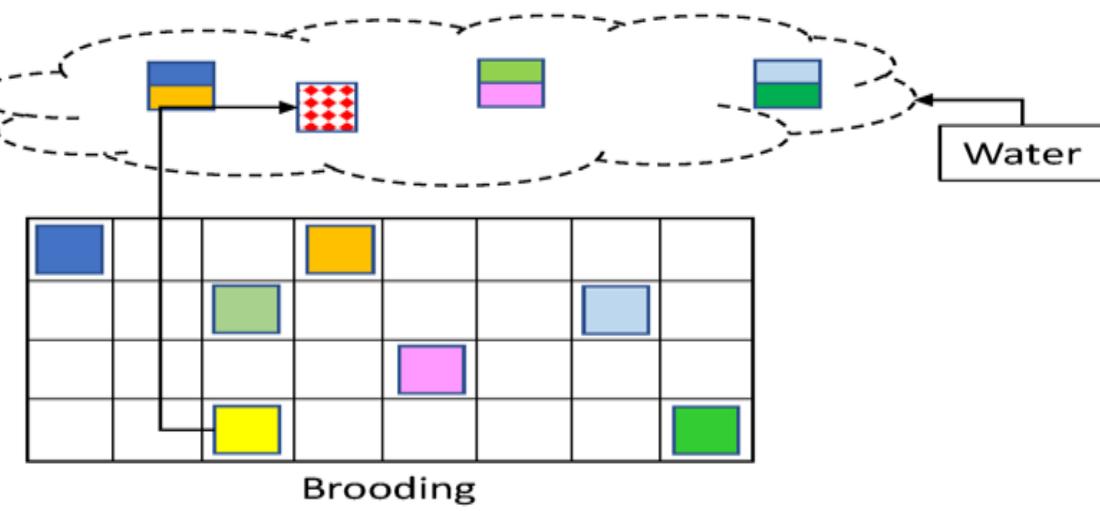
# Fases del algoritmo



1



2

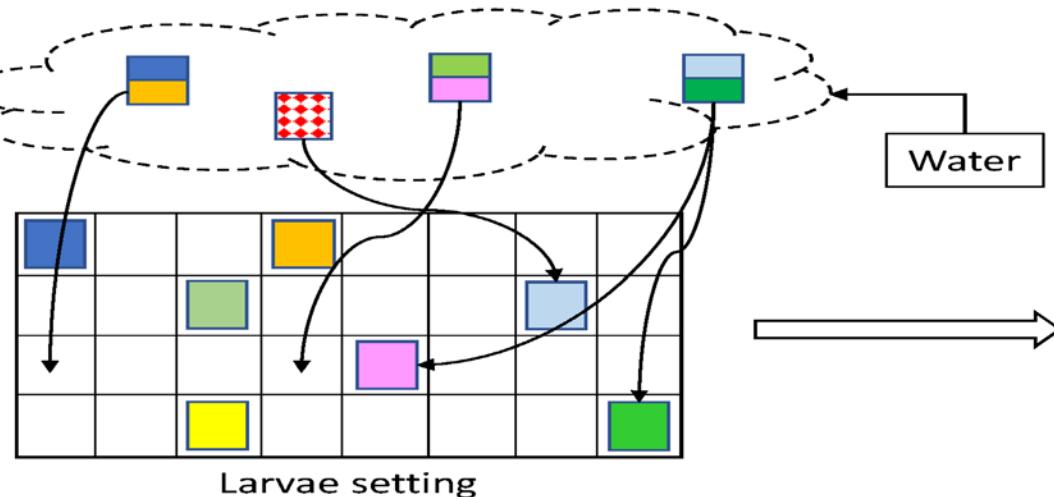


3

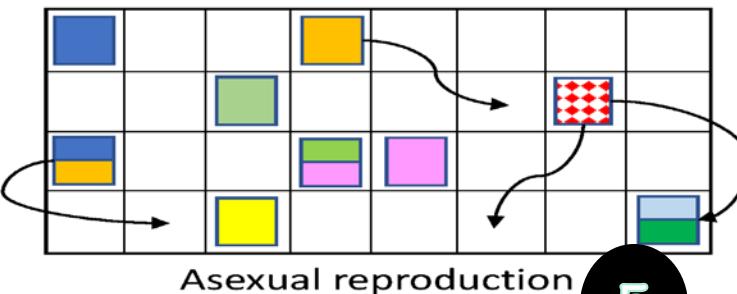
# Fases del algoritmo

Las larvas intentarán asentarse en un cuadrado aleatorio del arrecife

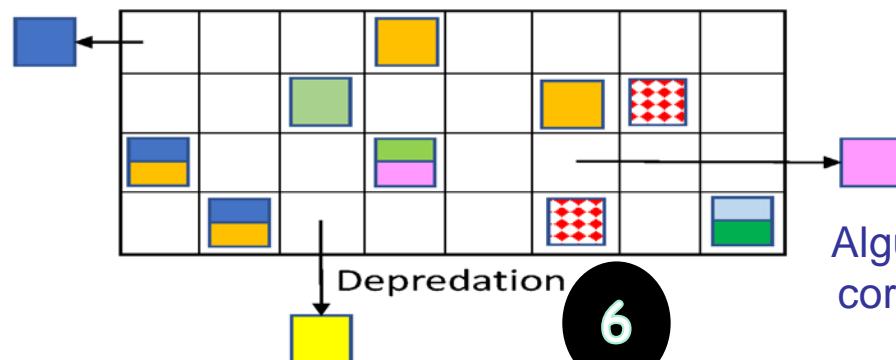
4



Los corales mas fuertes se duplican



5



6

Algunos corales mueren (por oros  
corales, mareas, otros animales..)

Fuente:

[http://agamenon.tsc.uah.es/Personales/sancho/CRO.html#:~:text=The%20Coral%20Reefs%20Optimization%20\(CRO,simulated%20to%20solve%20optimization%20problems.](http://agamenon.tsc.uah.es/Personales/sancho/CRO.html#:~:text=The%20Coral%20Reefs%20Optimization%20(CRO,simulated%20to%20solve%20optimization%20problems.)

# Algoritmo CRO

**Input** Reef dimensions  $m \times n$ , Initial occupation rate  $\rho_0$ , Fraction of broadcast spawning  $f_b$ , Fraction of asexual reproduction  $f_a$ , Predation fraction  $f_d$ , Predation probability  $p_d$

**Output** Solution with best *fitness*

```

1: procedure CRO
2:   reef  $\leftarrow$  initialization(n,m,ρ₀)
3:   repeat
4:     spawning_set, brooding_set  $\leftarrow$  partition(reef,fb)
5:     water  $\leftarrow$  broadcast_spawning(spawning_set)
6:     water  $\leftarrow$  {water, brooding(brooding_set)}
7:     water  $\leftarrow$  evaluation(water)
8:     reef  $\leftarrow$  larvae_setting(water, reef)
9:     reef  $\leftarrow$  asexual_reproduction(reef,fa)
10:    reef  $\leftarrow$  predation(reef,fd,pd)
11:   until stop_condition
12:   return best_solution(reef)
13: end procedure

```

Fuente:

[http://agamenon.tsc.uah.es/Personales/sancho/CRO.html#:~:text=The%20Coral%20Reefs%20Optimization%20\(CRO,simulated%20to%20solve%20optimization%20problems.](http://agamenon.tsc.uah.es/Personales/sancho/CRO.html#:~:text=The%20Coral%20Reefs%20Optimization%20(CRO,simulated%20to%20solve%20optimization%20problems.)

# Arte Evolutivo

- La idea es utilizar algoritmos evolutivos para hacer evolucionar las imágenes, música, etc, sobre una base estética, en lugar de una función de aptitud estricta.
- La función de aptitud la proporciona el usuario que selecciona sus individuos "favoritos" para la reproducción.



*Evolving Assemblages*, by Fernando  
Graça and Penousal Machado

1. Se genera una población aleatoria de imágenes
2. El usuario las evalúa y asigna un valor de aptitud
3. La selección se realiza según la aptitud o “calidad” : las imágenes con los valores de aptitud más altos tienen mayores probabilidades de ser seleccionadas para reproducción
4. El programa crea una nueva población de imágenes mediante el cruce y mutación de los individuos (imágenes) de la población actual

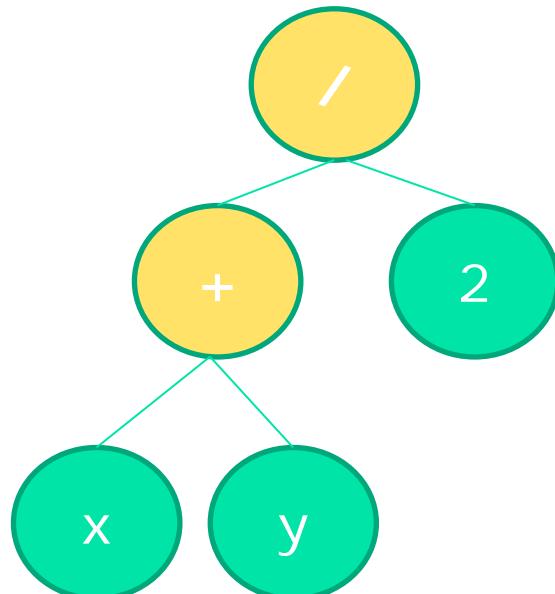
## Arte Evolutivo

- ❑ Los individuos son imágenes y el genotipo es un árbol con funciones y terminales.
- ❑ Se utilizan funciones simples: operaciones aritméticas, trigonométricas, lógicas, etc.
- ❑ El conjunto de terminales serán variables y constantes
- ❑ Las imágenes son representaciones gráficas de expresiones matemáticas.
- ❑ Cruce y mutaciones de las imágenes (árboles)

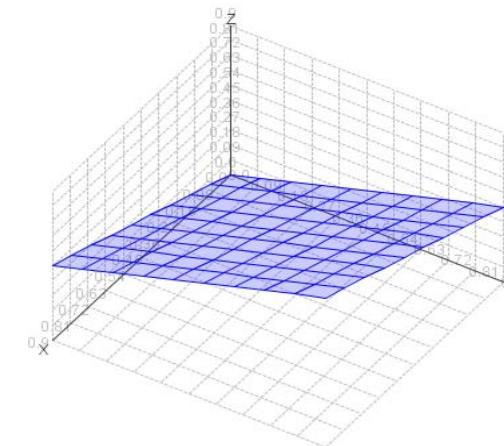
- La expresión

$$f(x) = (x + y) / 2$$

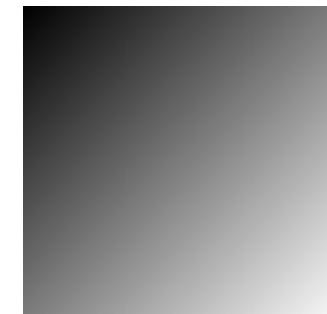
en formato de árbol



- Gráfica 3-d de la expresión  $f(x) = (x + y) / 2$



- Imagen generada por la asignación de un valor de escala de grises a cada valor de  $f(x)$ .



# Acerca de Creative Commons



## □ Licencia CC ([Creative Commons](#))

Este tipo de licencias ofrecen algunos derechos a terceras personas bajo ciertas condiciones.

Este documento tiene establecidas las siguientes:



### Reconocimiento (*Attribution*):

En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.



### No comercial (*Non commercial*):

La explotación de la obra queda limitada a usos no comerciales.



### Compartir igual (*Share alike*):

La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

Pulsa en la imagen de arriba a la derecha para saber más.