Programación Evolutiva

Tema 8: Extensiones de los algoritmos genéticos. Otros temas

Carlos Cervigón, Lourdes Araujo. 2015-2016

Universidad Complutense Madrid

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
  - Arte Evolutivo

Universidad Complutense Madrid

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.

Algoritmos Genéticos con edades

Aparte del parecido con la naturaleza encontramos las siguientes motivaciones:

- Mantener el control de la edad es suficiente para controlar la presión selectiva, haciendo innecesaria la selección de criadores.
- 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 ρ es el coeficiente de reproducción (valores próximos a 0,4).
- La calidad del AG mejora al aumentar ρ 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

Universidad Complutense Madrid

Algoritmos Genéticos con edades

- Lo más complejo es la asignación de duraciones:
- 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.

Algoritmos Genéticos con edades

3 estrategias:

- Asignación proporcional

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

- 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

$$TiempoVida(\mathbf{v}) =$$

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

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.

Universidad Complutense Madrid

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.

Universidad Complutense Madrid

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.

Universidad Complutense Madrid

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→A) y heterocigotos (Aa→A).
  - Las características recesivas solamente se expresan en homocigotos (aa→a).

Búsqueda multiobjetivo

- En muchos problemas se necesita optimizar simultáneamente varios objetivos $f_1(x), \ldots, f_r(x)$
- Matemáticamente:

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

- El óptimo global ya no es inmediato:
  - ¿Optimizar una función sacrificando el resto, la media de todas las funciones ...?
- ***Óptimo de Pareto***: una solución $x \in X$ es un óptimo de Pareto cuando **no** existe ninguna solución mejor en ningún objetivo: soluciones no dominadas

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:
  - Son los factores de ponderación, que por convenio la suma de los factores de ponderación vale 1.
  - La elección de los w_k es fundamental

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

Universidad Complutense Madrid

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 párametros en mercados financieros*, etc.

Universidad Complutense Madrid

Búsqueda multiobjetivo

- Usualmente los objetivos no pueden reducirse a una base común.
- Técnica de las subpoblaciones:
  - Se utiliza una población dividida en subpoblaciones de igual tamaño, cada una de ellas responsable de un único objetivo.
  - La selección se realiza dentro de cada subpoblación.
  - El cruce se realizaba entre subpoblaciones.
  - Resultados mediocres.
- Los mejores resultados se obtienen con la búsqueda de la soluciones no dominadas.

Búsqueda multiobjetivo

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

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

tal que

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

- Se espera que las mejores soluciones estén entre ellas.

Búsqueda multiobjetivo

- Uso de las soluciones no dominadas:
  - Se ordenan los individuos en función de su grado de dominio en la población: a los individuos no dominados se les da el primer puesto, el segundo se da a los individuos no dominados por otros individuos más que por los del primer puesto, etc.
  - Esos índices se usan como aptitudes brutas de los respectivos individuos
  - Las técnicas de escisión se introducen para mantener un alto grado de diversidad a lo largo de todo el proceso.

Algoritmos Evolutivos Paralelos

Paralelización Global

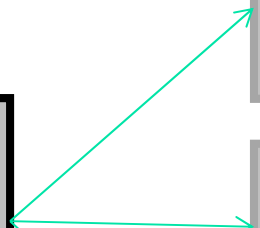
(Misma estructura del AG)

1.Ejecución de varios AGs en Paralelo

2.Sólo la Evaluación en paralelo

Subpoblaciones que evolucionan por separado

(Cambia la estructura del AG)



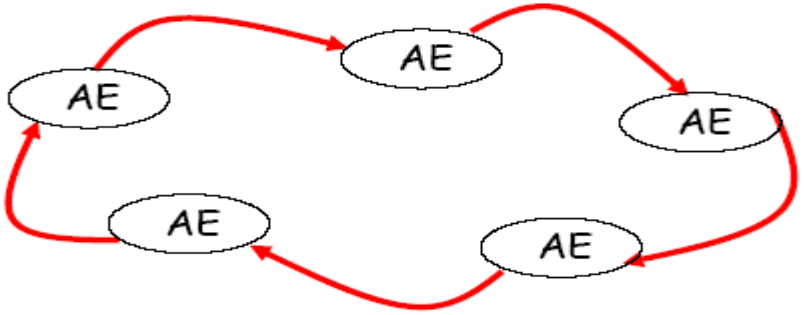
Grano Grueso (Islas)

Grano Fino (AC)

Híbrido (grano fino y grueso)

Modelos distribuidos o en isla

- Se divide la población en subpoblaciones mas pequeñas que evolucionan independientemente en cada procesador o isla.
- Un conjunto de subpoblaciones evolucionan independientemente en cada procesador o isla.
- En cada procesador se lleva a cabo un Algoritmo Evolutivo
- 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



Universidad Complutense Madrid

Modelos distribuidos o en isla

- 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

Universidad Complutense Madrid

Modelos distribuidos o en isla

- ¿Cada cuánto tiempo debemos intercambiar individuos?
  - Demasiado rápido y todas las poblaciones convergen hacía 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

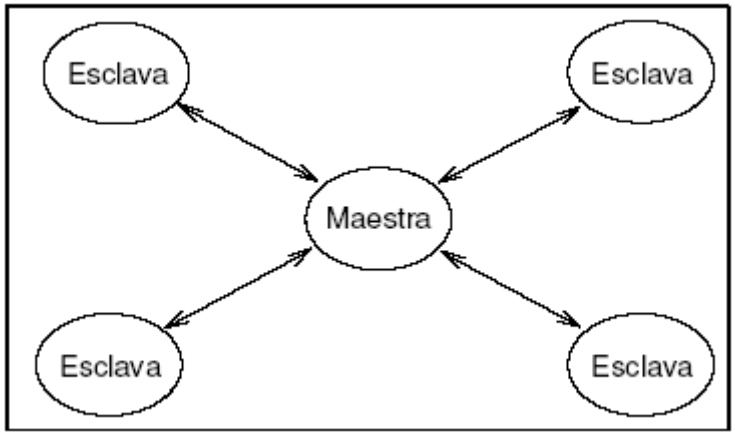
Universidad Complutense Madrid

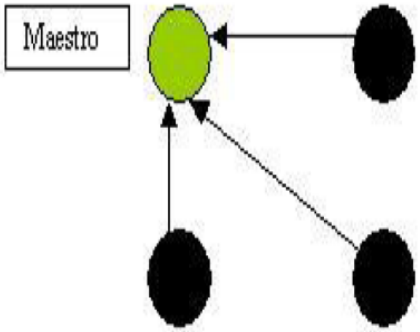
Modelos distribuidos o en isla

- Se pueden distinguir diferentes modelos de islas en función de la comunicación entre las subpoblaciones.
- Modelos típicos:
  - Estrella
  - En red
  - En anillo

Modelo en estrella: maestro-esclavo

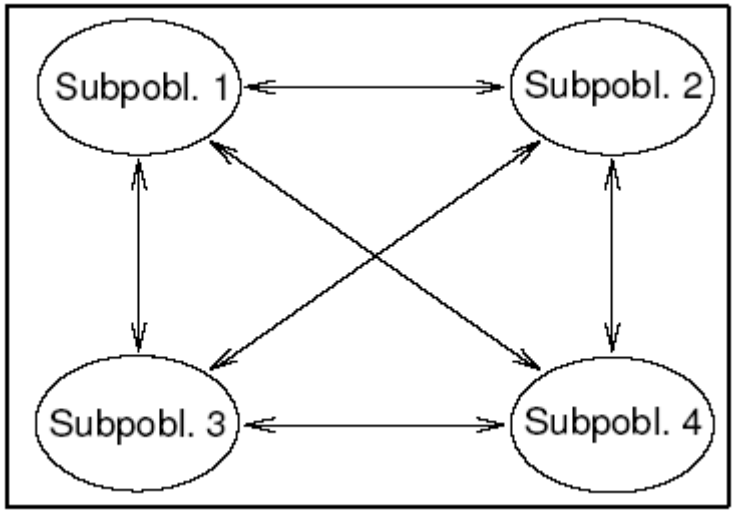
- Una subpoblación se seleccionada como maestra (la que tiene mejor media en el valor de la función objetivo), siendo las demás consideradas como esclavas.
- Todas las subpoblaciones esclavas mandan sus ***h1*** mejores individuos (***h1*** **>1**) a la subpoblación maestra la cual a su vez manda sus ***h2*** mejores individuos (***h2*** **>1**) a cada una de las subpoblaciones esclavas.





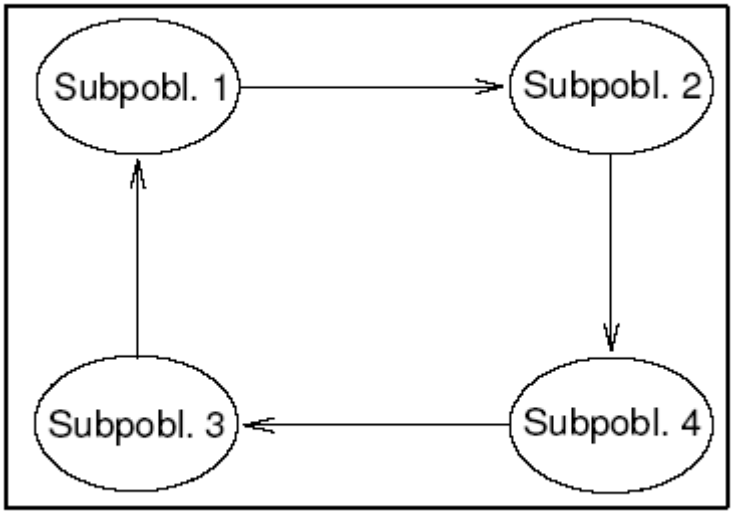
Modelo en red: todos con todos

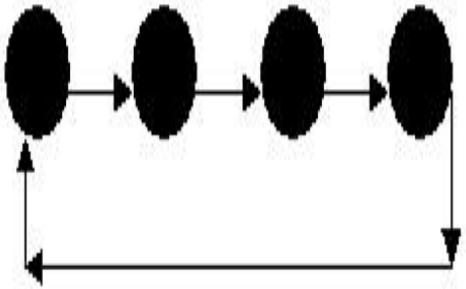
- No hay una jerarquía entre las subpoblaciones; todas mandan sus ***h3*** (***h3>1***) mejores individuos al resto de las subpoblaciones.



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.



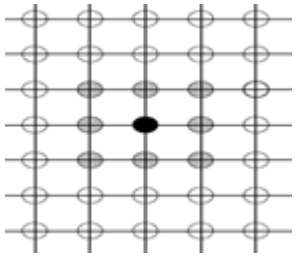


Algoritmo

```
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);
}
```

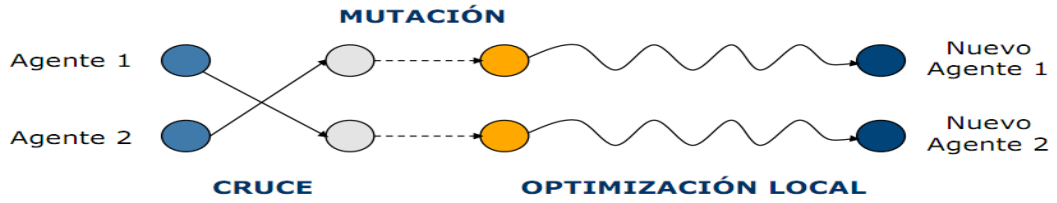
Grano fino: modelo celular

- 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.
  - La evaluación se realiza simultáneamente para todos los individuos
  - La selección, reproducción y cruce 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.
    - aislamiento por distancia



Algoritmos meméticos

- **Memes**: unidades de información cultural (que se pueden transmitir entre individuos), que engloban los conceptos, las modas y las tradiciones de una sociedad. Rol análogo a los genes en evolución cultural. (Dawkins, R. 1979 The Selfish Gene)
  - Algoritmo híbrido con operadores de búsqueda local que permite a los individuos "cambiar" antes de crear la nueva población.
  - Los individuos pueden copiar o imitar "parte" de los genes de otros individuos para mejorar su fitness
  - Nuevos operadores de búsqueda o incluir la información memética en los operadores tradicionales de cruce y mutación
  - Lamarckiano, Baldwiniano



Universidad Complutense Madrid

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...

Optimización basada en grupos de partículas: PSO

- 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.

Universidad Complutense Madrid

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*rand()*(pBest[]-pos[]) + c2*rand()*(gBest[]-pos[])

Actualizar la posición según ecuación

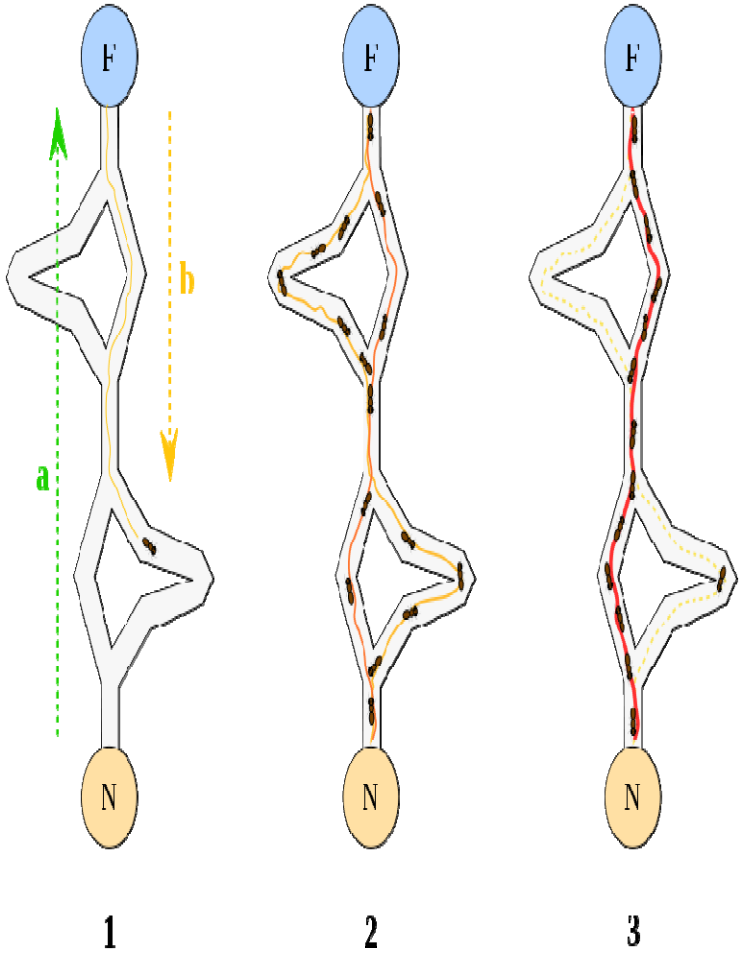
pos[] = pos[] + v[]

While no fin (iteraciones o error)

Optimización basada en colonias de hormigas: ACO

- 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

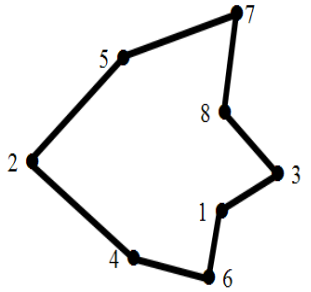
http://en.wikipedia.org/wiki/Ant_colony_optimization.



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:

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



- 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*.
- α y β son dos parámetros del algoritmo, los cuales hay que ajustar.

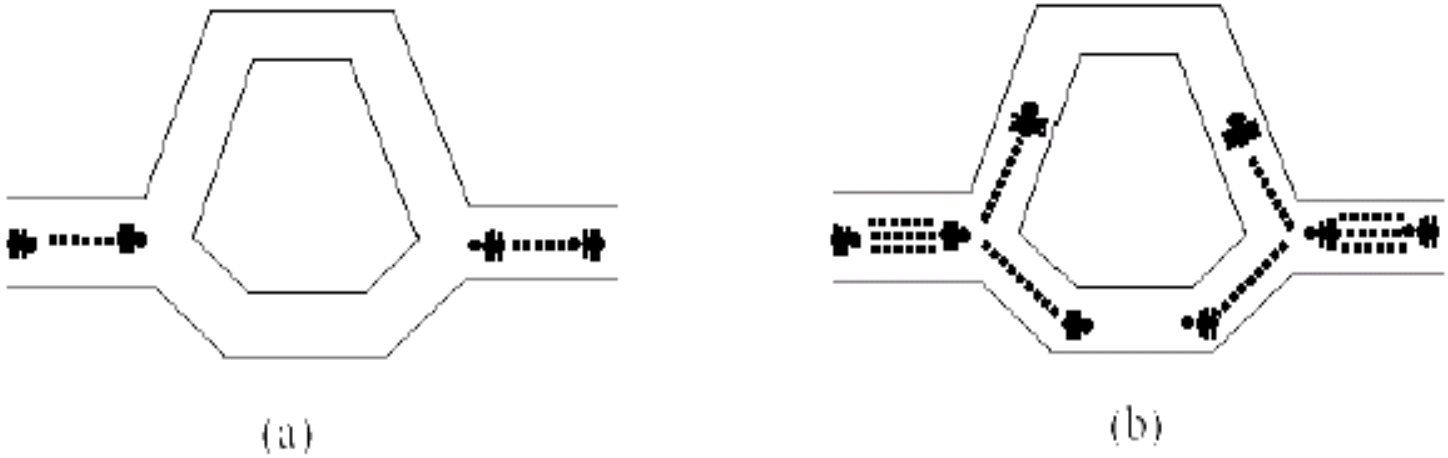
Universidad Complutense Madrid

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

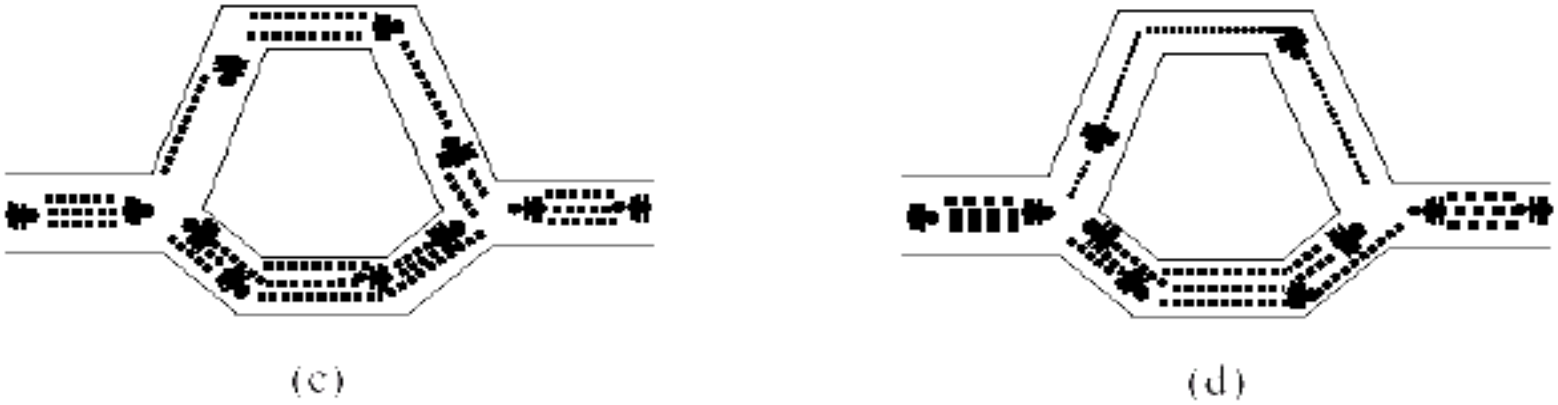
- En las siguientes figuras se observa como las hormigas establecen el camino más corto.
- En la figura (a) las hormigas llegan al punto en que tienen que decidir por uno de los caminos que se les presenta.



- 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.



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 feromona artificial y por una información calculada a priori de manera heurística. La regla probabilística para el caso del TSP es:

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

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 feromona acumulada sobre el arco *(i,j)* de la red en la iteración *t*.
- η_{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*.
- α y β 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:

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

- donde ρ 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.

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

Universidad Complutense Madrid

Arte Evolutivo

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

Universidad Complutense Madrid

Arte Evolutivo

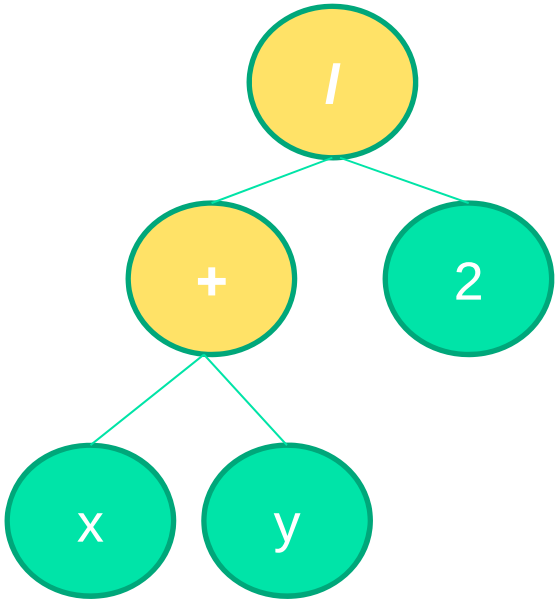
- Los individuos son imágenes y el genotipo es un árbol con funciones y terminales.
- Se utilizan funciones simples: operaciones aritméticas, trigonométricas, lógicas, etc.
- El conjunto de terminales serán variables y constantes
- La imágenes son representaciones gráficas de expresiones matemáticas.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

Arte Evolutivo

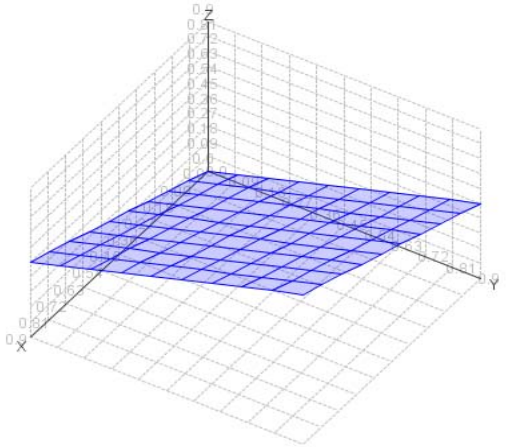
- La expresión

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

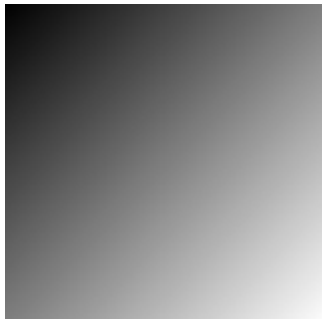
en formato de árbol



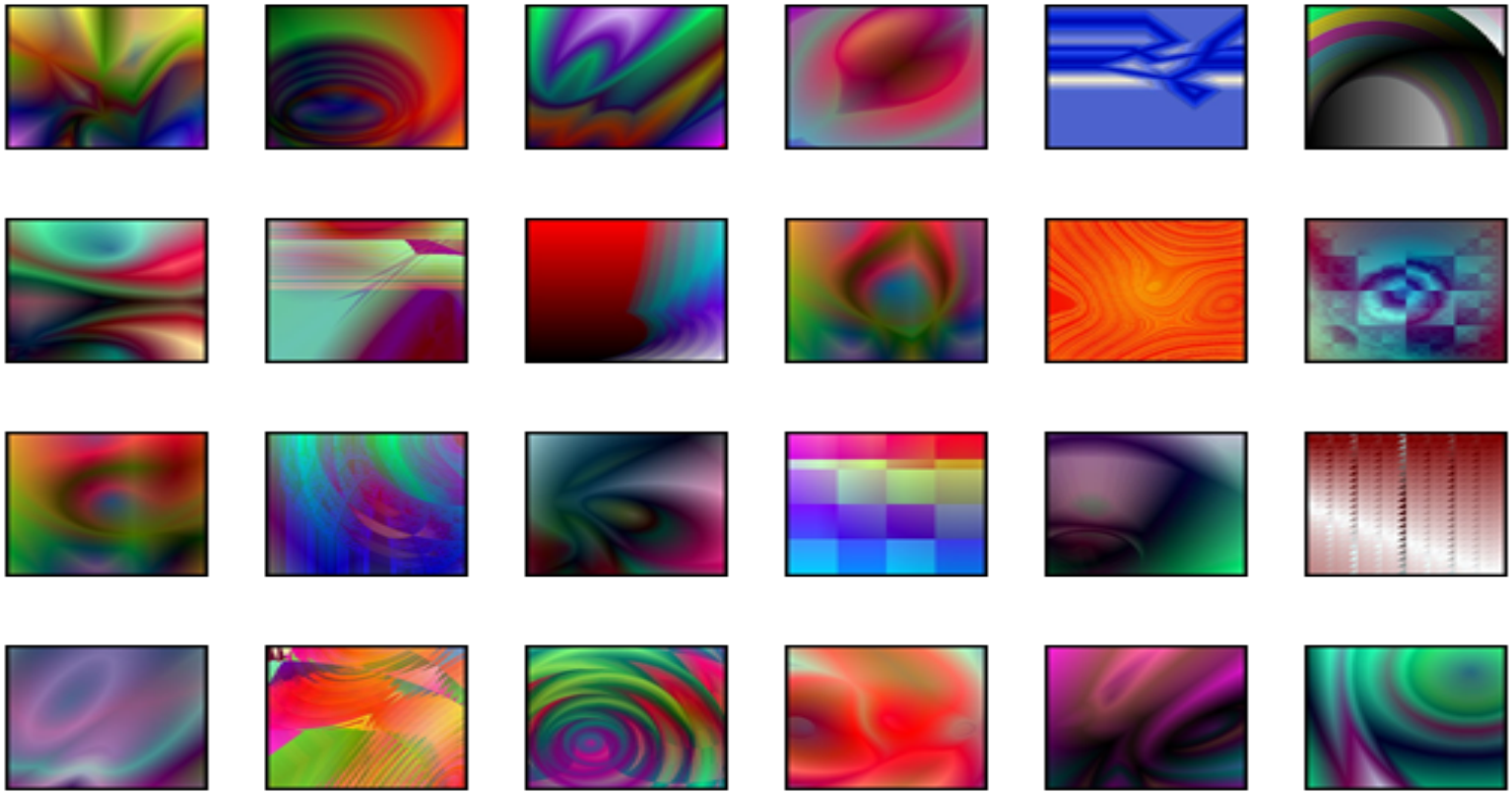
- Gráfica 3-d de la expresión `f (x) = (x + y) / 2`



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



Arte Evolutivo : ejemplos



www.Picbreeder.org

www.endlessforms.com