Seminario Permanente de Formación en Inteligencia Artificial Aplicada a la Defensa

Optimización Inteligente II

Daniel Molina Cabrera

Instituto Andaluz de Investigación en Data Science and Computational Intelligence (DaSCI)

Dpto. Ciencias de la Computación e I.A.

Universidad de Granada

dmolina@decsai.ugr.es

http://sci2s.ugr.es

DaSCI

UNIVERSIDAD DE GRANADA

La semana pasada vimos

1. Concepto de Optimización y Metaheurísticas
2. Ejemplos Reales de uso de Metaheurísticas
3. Algoritmos Evolutivos y Genéticos
4. Caso de Estudio (Aplicando el AG para optimizar)

Optimización Inteligente (II)

1. Swarm Intelligence

2. Tipos de Optimización

3. *Frameworks* de Algoritmos Evolutivos

4. Caso de Estudio (Problemas continuos)

Optimización Inteligente (II)

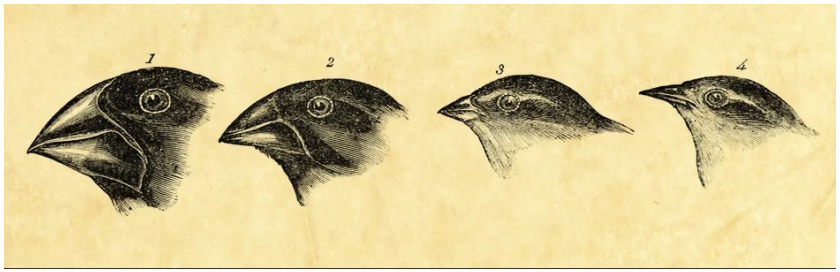
1. Swarm Intelligence

2. Tipos de Optimización

3. *Frameworks* de Algoritmos Evolutivos

4. Caso de Estudio (Problemas continuos)

Algoritmo Genético vs Swarm Intelligence



Se crean nuevos individuos mediante evolución

Individuos sencillos que se comunican para resolver problemas complejos

Swarm Intelligence

"La inteligencia colectiva emergente de un grupo de agentes simples"

"The emergent collective intelligence of groups of simple agents"

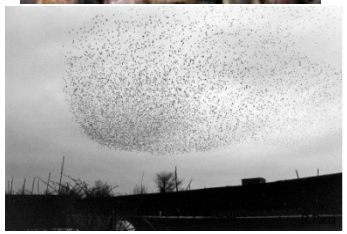
"**Algoritmos o mecanismos distribuidos de resolución de problemas inspirados en el comportamiento colectivo de colonias de insectos sociales u otras sociedades de animales".**

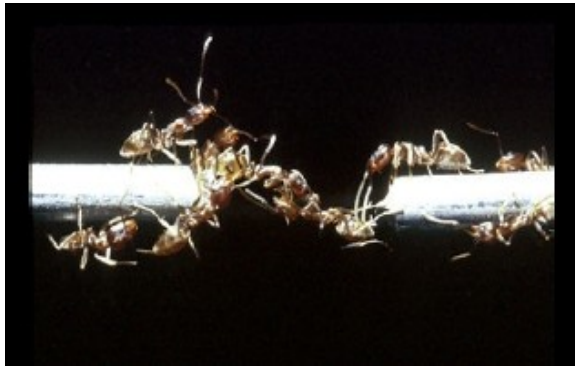
(Bonabeau, Dorigo, Theraulaz, 1999)

E. Bonabeau, M. Dorigo, G. Theraulaz
Swarm Intelligence. From Nature to Artificial Systems.
Oxford University Press, 1999.

Inspiración *Swarm Intelligence*

"Dumb parts, properly connected into a swarm, yield to smart results"





sociedades de insectos
(bees, wasps, ants, termites)
(abejas, avispas, hormigas, termitas)

schools of fish
(bancos de peces)

herds of mammals
(manadas de mamíferos)

Insectos Sociales

Insectos sociales: La complejidad y la sofisticación de auto-organización se lleva a cabo sin un líder claro

Los modelos de las colonias/sociedades de insectos por medio de sistemas auto-organizativos puede ayudar al diseño de sistemas artificiales distribuidos para la resolución de problemas.

De sociedades de insectos a Sistemas de Enjambre

Comportamiento emergente

- Las colonias de insectos llevan a cabo actuaciones de nivel complejo de forma inteligente, flexible y fiable, actuaciones que no serían factibles si tuviesen que ser realizadas por un insecto de forma individual (éstos son no inteligentes, no fiables, simples).

- **Los insectos siguen reglas simples, y utilizan comunicación local simple**

- La estructura global (nido) emerge desde las acciones de los insectos (las cuales son no fiables atendidas individualmente)

SI: Abejas

Abejas

- Cooperación de la colmena
- Regulan la temperatura de la colmena
- Eficiencia vía especialización: división de la labor en la colonia
- Comunicación: Las fuentes de comida son explotadas de acuerdo a la calidad y distancia desde la colmena

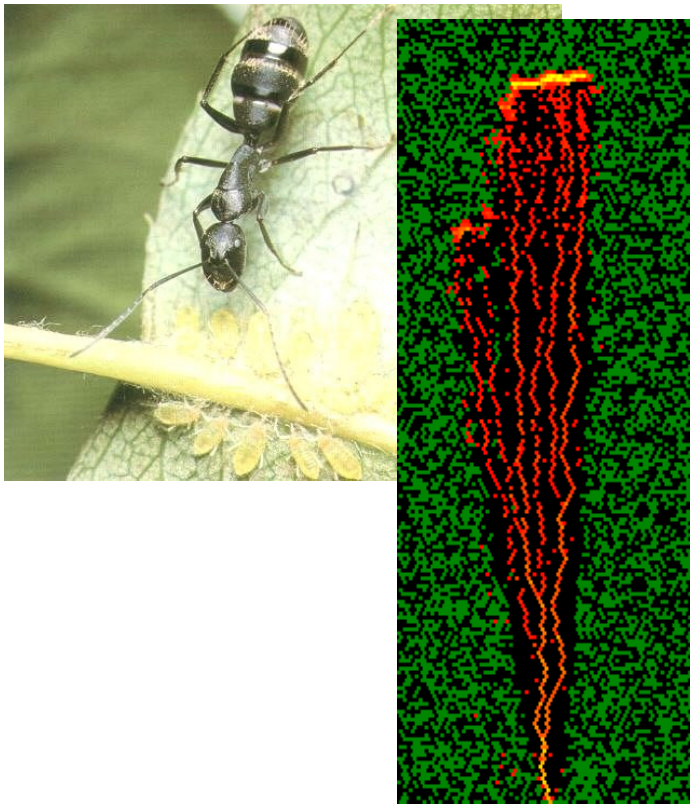
Ejemplos

Termitas

- Nido con forma de cono con paredes externas y conductos de ventilación
- Camaras de camadas en el centro de la colmena
- Rejillas del ventilación en espirales
- Columnas de soporte

Ejemplos

Hormigas



- Organizan autopistas “hacia y desde” la comida por medio de rastros de feromona (pheromone)

Algoritmos de *Swarm Intelligence* más populares

- Ant colony optimization – ACO

  Optimización basada en colonias de hormigas

  - **Conjunto de técnicas inspiradas por las actividades de una colonia de hormigas**

- Bee colony optimization – BCO

  Optimización basada en colonias de abejas

  - **Conjunto de técnicas inspiradas por las actividades de una colonia de abejas**

- Particle swarm optimization – PSO

  Optimización basada en nubes de partículas

  - **Conjunto de técnicas inspiradas en el comportamiento de las bandadas de aves o bancos de peces**

Colonias de Hormigas

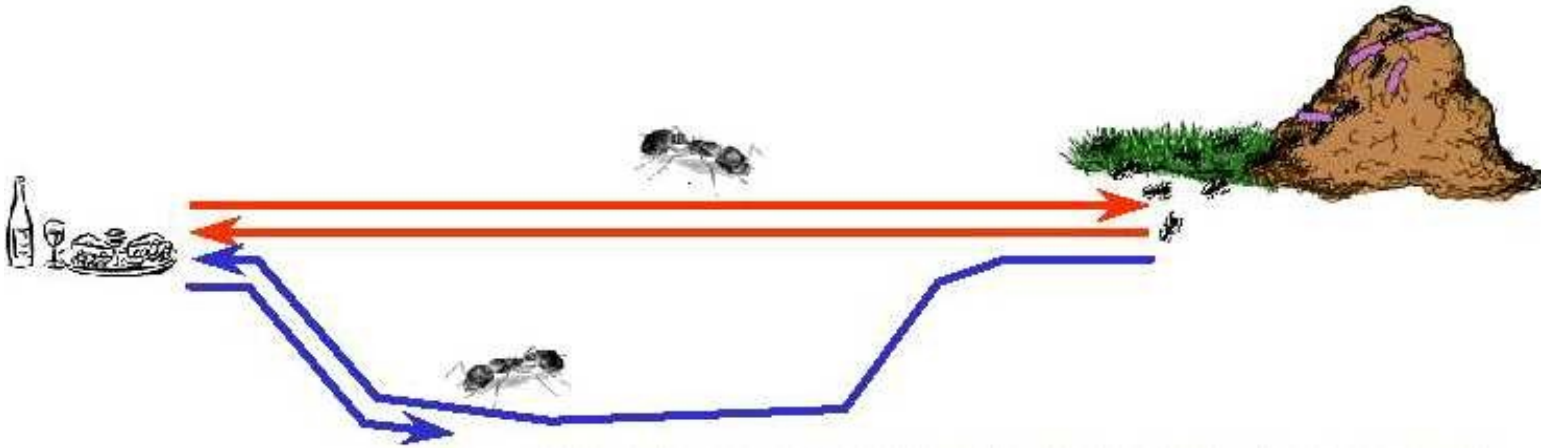
- Inspirado en la obtención del camino usando *feromonas*.

Rastro de feromonas

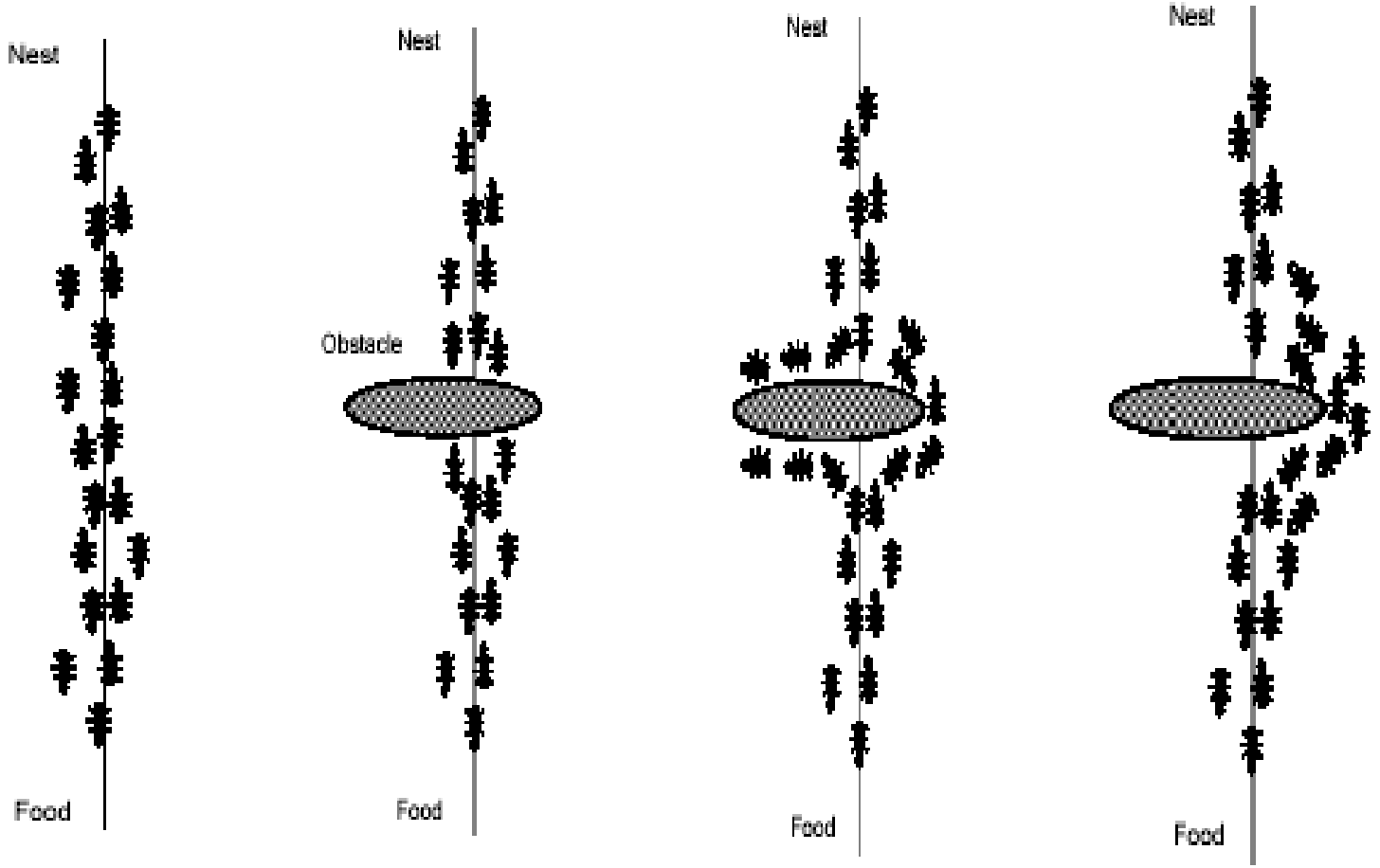
¿Optimizan el camino?

Ant Colony Optimization

- La analogía más cercana a ACO son los problemas de rutas en grafos.
- Mientras las hormigas buscan comida, depositan rastros de feromona que atraen a otras hormigas. Desarrollan caminos mínimos entre la comida y el hormiguero.



Ant Colony Optimization: Comportamiento natural



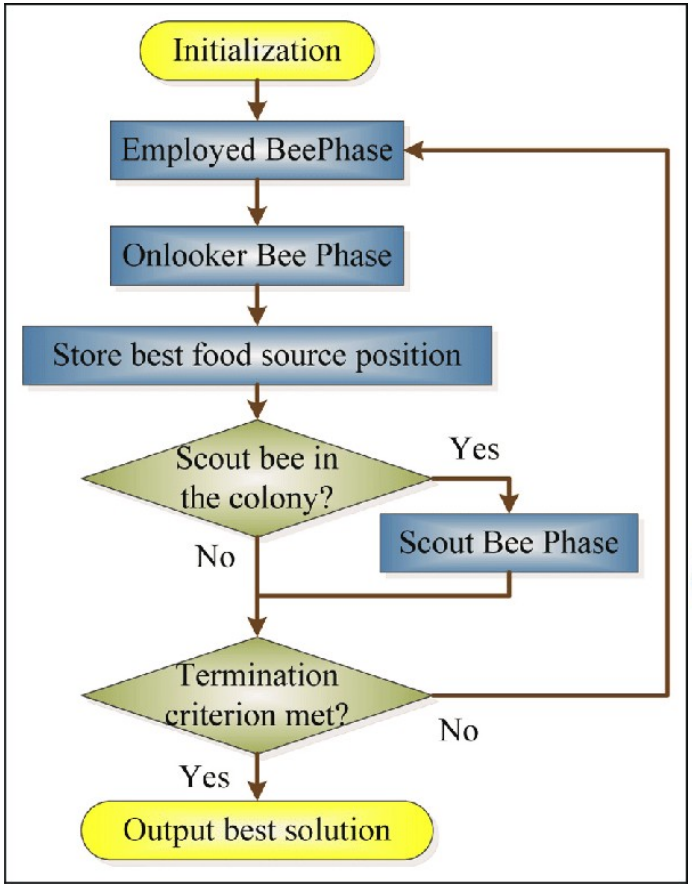
Aplicando hormigas al TSP

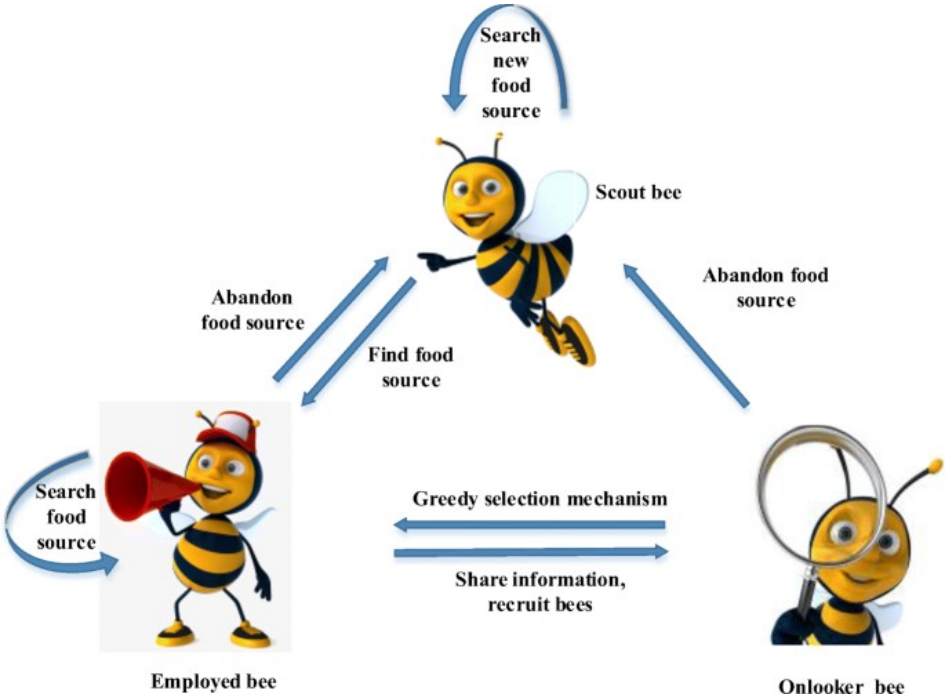
- Líneas rojas: rastro de feromonas
- Obtiene muy buenos resultados.

Bee Colony Optimization

- **Los algoritmos de optimización basados en abejas están basados en la recogida de comida (nectar) de las abejas.**

Artificial Bee Colony



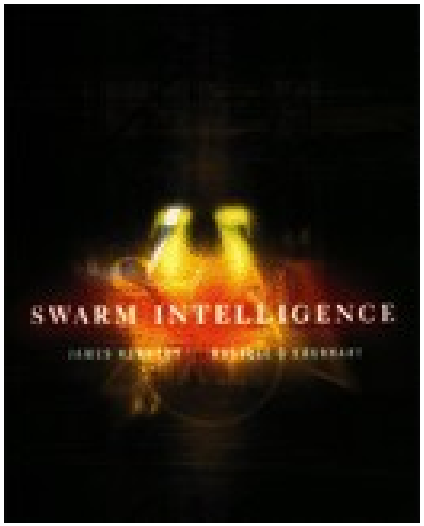


Artificial Bee Colony

1. Inicialización: Soluciones aleatorias.
2. Exploring: Soluciones intermedias entre pares aleatorios, se queda con el mejor (*exploring*).
3. Si la solución no mejora, incrementa contador.
4. Calcular fitness de cada solución.
5. Con una ruleta basada en el fitness, mover hacia otra solución (*onlooker*, alrededor de las mejores).
6. Si alguna solución no mejora durante N iteraciones, reinicia (*scout*).

Sistemas de Partículas (PSO)

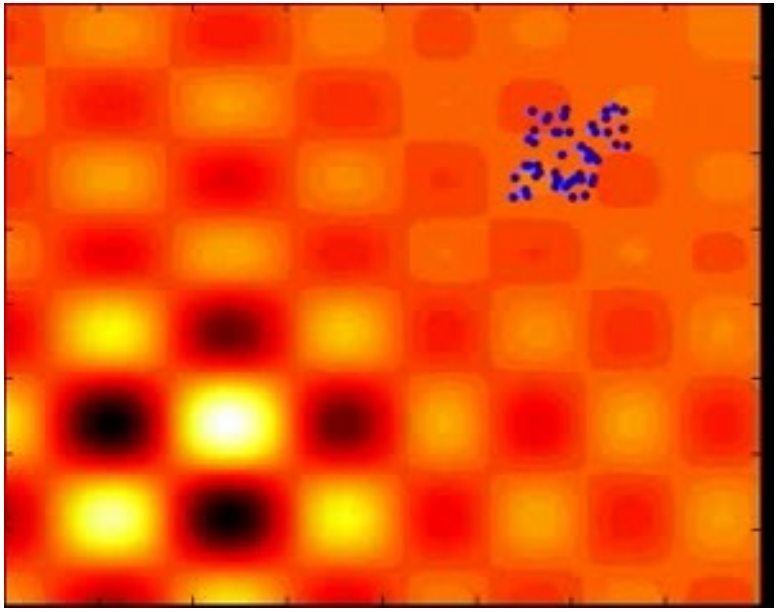
Inspirados en el comportamiento social de bandadas de aves o peces.



Kennedy, J., Eberhart, R. C., and Shi, Y.,
Swarm intelligence
San Francisco: Morgan Kaufmann Publishers, 2001.

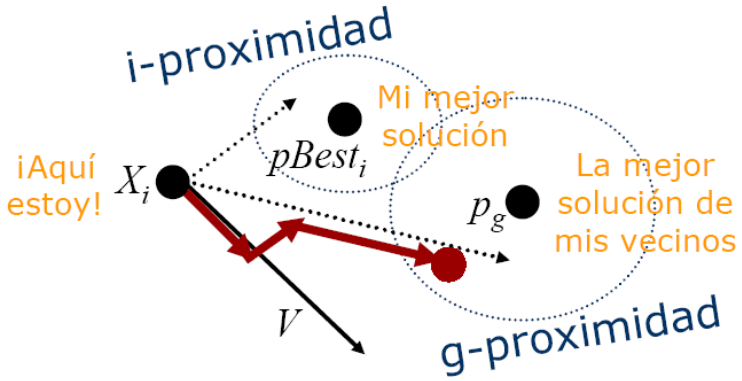
Particle Swarm Optimization

- PSO simula el comportamiento de las bandadas de aves.
- Cada solución (partícula) es un “ave” en el espacio de búsqueda que está siempre en continuo movimiento y que nunca muere.



Partículas en PSO

- Cada partícula tiene un fitness, una posición y un vector velocidad que dirige su “vuelo”. El movimiento de las partículas por el espacio está guiado por las partículas óptimas en el momento actual.



Inicialización del PSO

- La nube se inicializa generando las posiciones y las velocidades iniciales de las partículas.
- Las posiciones se pueden generar aleatoriamente en el espacio de búsqueda, de forma regular, o con una combinación de ambas.
- Las velocidades se generan aleatoriamente, con cada componente en el intervalo [-Vmax, Vmax].
  - No es conveniente fijarlas a cero, no se obtienen buenos resultados.
  - Vmax será la velocidad máxima que pueda tomar una partícula en cada movimiento.

Movimiento de las partículas

- El primer paso es ajustar el vector velocidad, para después sumárselo al vector posición.
- Las fórmulas empleadas son:

$$v_{id} = v_{id} + \phi_1 \cdot rnd() \cdot (pBest_{id} - x_{id}) + \phi_2 \cdot rnd() \cdot (g_{id} - x_{id})$$

COGNITIVO **SOCIAL**

$$x_{id} = x_{id} + v_{id}$$

  - *xi* es la partícula actual, *vi* su velocidad.
  - φ_1, φ_2 son ratios de aprendizaje (pesos) que controlan los componentes cognitivo y social,

Movimiento de las partículas

- Para evitar que v se incremente, se usa factor de inercia:

$$v_{id} = \omega \cdot v_{id} + \varphi_1 \cdot rnd() \cdot (pBest_{id} - x_{id}) + \varphi_2 \cdot rnd() \cdot (lBest_{id} - x_{id})$$

  donde ω se inicializa a 1.0 y se va reduciendo gradualmente a lo largo del tiempo (medido en iteraciones del algoritmo).

- El valor ω debe mantenerse entre 0.9 y 1.2. Valores altos provocan una búsqueda global (diversificación) y valores bajos una búsqueda más localizada (más intensificación).

Esquema del PSO

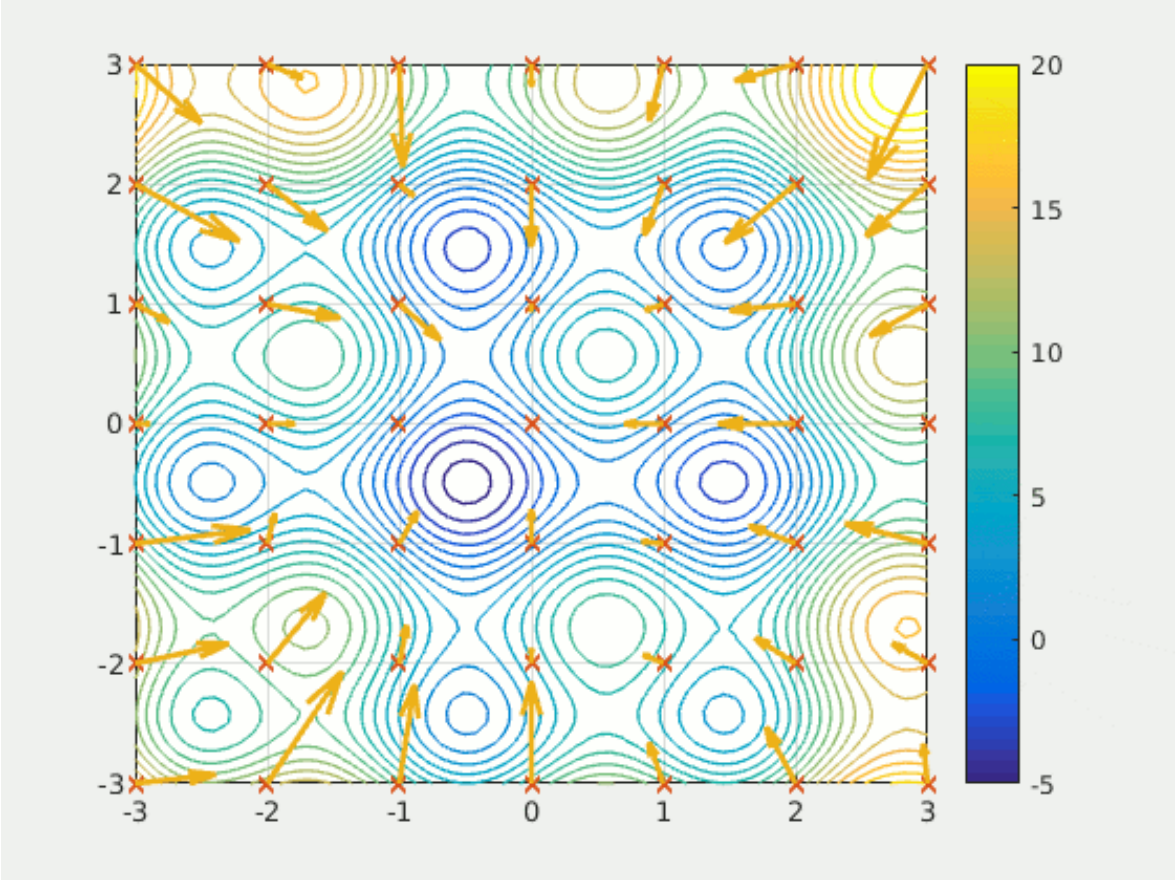
t = 0;
Para i=1 hasta Número_partículas
inicializar X_i y V_i;

Mientras (no se cumpla la condición de parada) hacer
$t \leftarrow t + 1$
Para i=1 hasta Número_partículas
evaluar X_i;
Si $F(X_i)$ es mejor que $F(pBest)$ entonces
$pBest_i \leftarrow X_i$; $F(pBest_i) \leftarrow F(X_i)$
Si $F(pBest)$ es mejor que $F(gBest)$ entonces
$gBest \leftarrow pBest_i$; $F(gBest_i) \leftarrow F(pBest_i)$

Para i=1 hasta Número_partículas
Calcular V_i, la velocidad de X_i, de acuerdo a $pBest_i$ y $gBest_i$
Calcular la nueva posición X_i, de acuerdo a X_i y V_i

Devolver la mejor solución encontrada

Ejemplo Movimiento



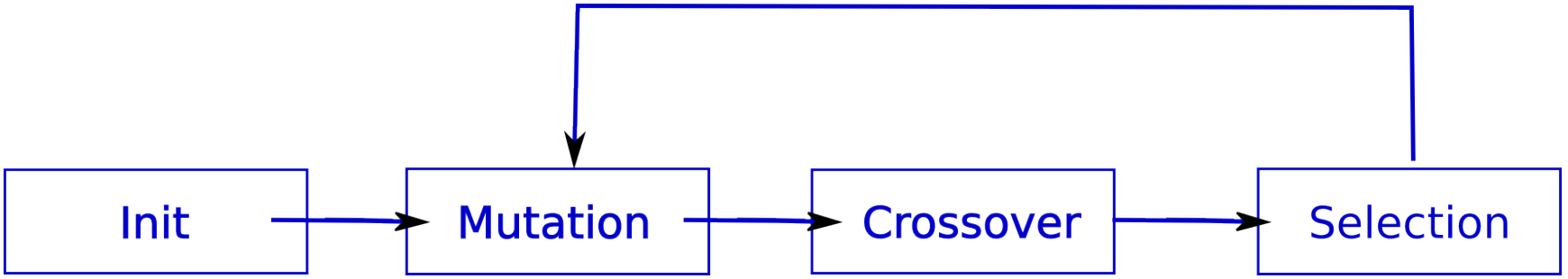
Differential Evolution

- Modelo evolutivo que enfatiza la mutación, utiliza un operador de cruce/recombinación a posteriori de la mutación.
- Propuesto para optimización con parámetros reales por R. Storn, 1997.

R. Storn, Differential Evolution, A simple and efficiente heuristic strategy for global optimization over continuous spaces. Journal of Global Optimization, 11 (1997) 341-359.

Kenneth V. Price, Rainer M. Storn, and Jouni A. Lampinen
Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series)
Springer-Verlag, 2005.

Esquema del DE

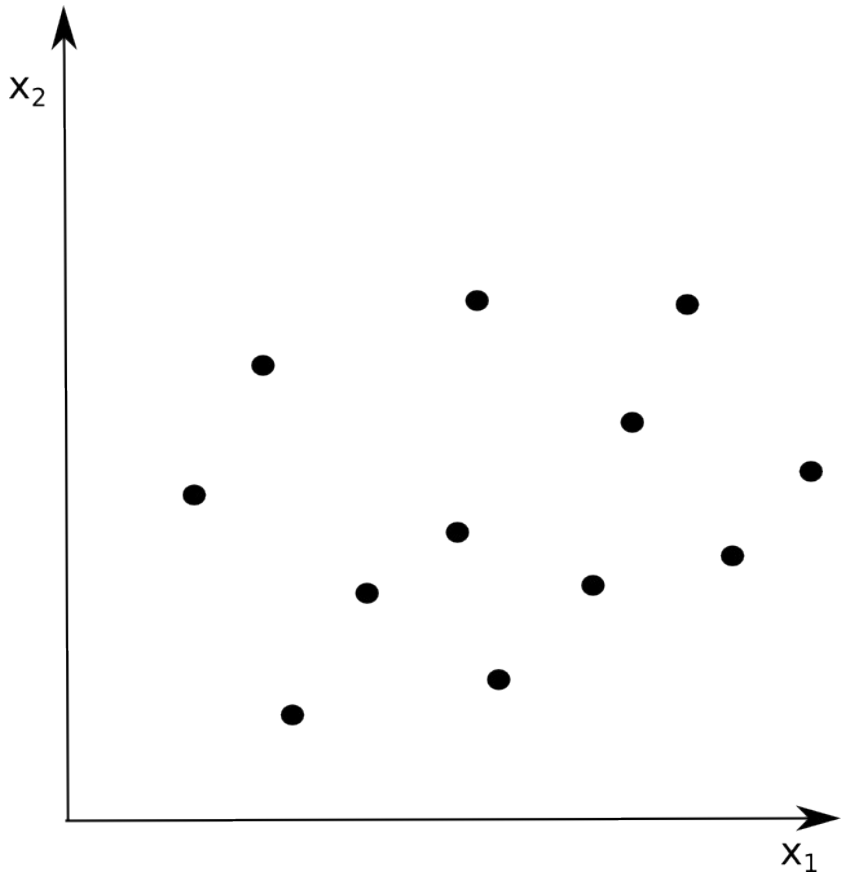


- *Init*: Inicia la población de soluciones.
- *Mutation:* Crea nueva solución a partir de actuales.
- *Crossover:* Combina antigua y nueva.
- *Selection:* Selecciona cual se mantiene en la población.

Mutación

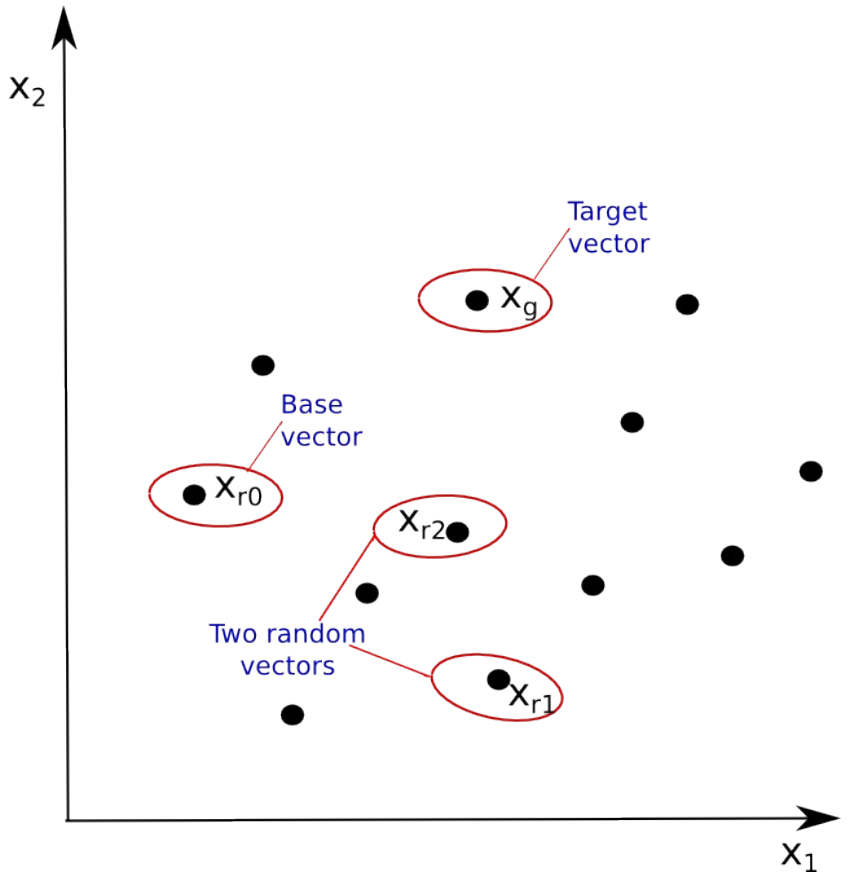
- Se genera por cada solución x_i una nueva solución combinando otras dos soluciones.
  - Se calcula vector diferencia entre ambas.
  - Se usa ese vector para *mover* la solución.
- Dos enfoques:
  - Escoger soluciones aleatorias.
  - Valorar vector distancia hacia el mejor.

Ejemplo de Mutación



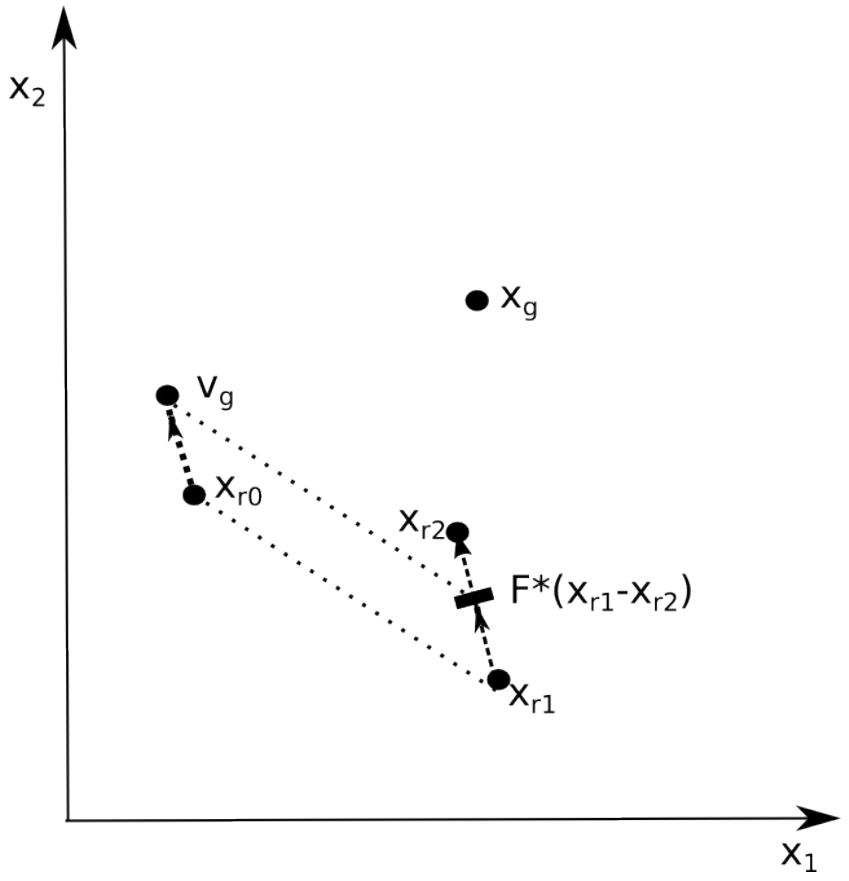
- Población inicial

Ejemplo de Mutación



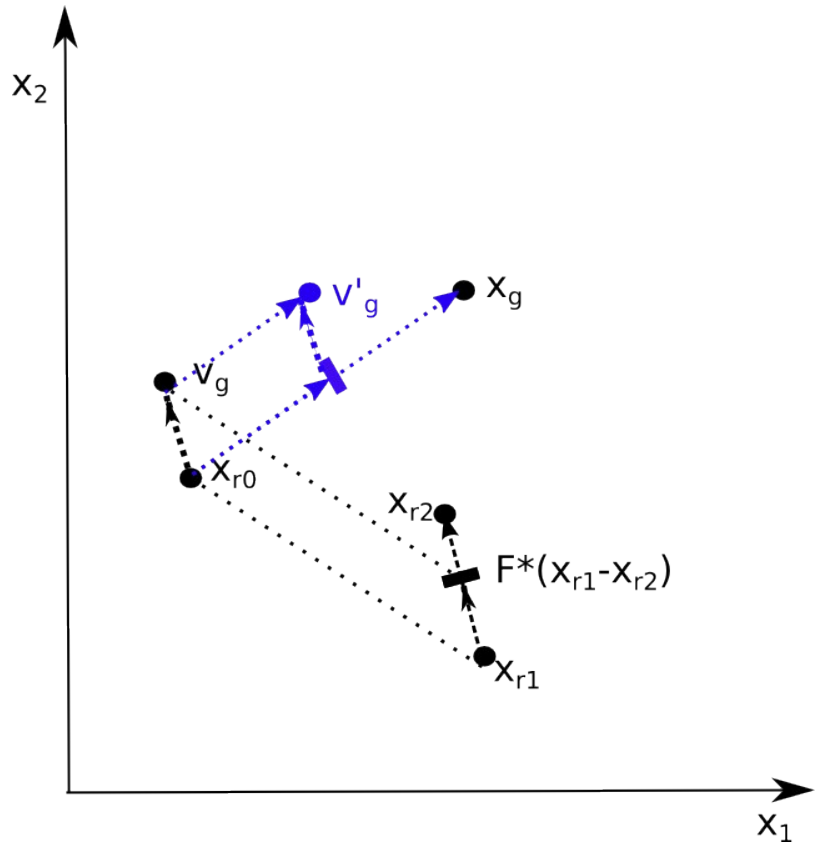
- Selecciona elementos aleatorios.

Ejemplo de Mutación



- Calcula vector distancia.
- Utiliza % del vector para crear nueva solución.
  - F = porcentaje.

Ejemplo de Mutación



- Tambien puede usar la dirección hacia la mejor solución.

Estrategias de mutación

- Mutación clásica DE/bin/1:

$$V_{i,g+1} = X_{i,g} + r_1 * F * \left(X_{r2,g} - X_{r1,g} \right)$$

- Mutación DE/best/1:

$$V_{i,g+1} = X_{best,g} + r_1 * F * \left(X_{r2,g} - X_{r1,g} \right)$$

- Mutación DE/current-to-best/:

$$V_{i,g+1} = X_{i,g} + r_1 * F * \left(X_{best} - X_{i,g} \right) + r_2 * F * \left(X_{r2,g} - X_{r1,g} \right)$$

Cruce y selección

- Recombinación clásica:

Con respecto a cada vector objetivo $x_{i,g}$ en la población actual, un nuevo vector $u_{i,g}$ se genera cruzando el vector objetivo $x_{i,g}$ con el correspondiente vector mutado $v_{i,g}$ bajo un ratio predefinido de cruce Cr ∈ [0, 1].

$$u_{j,i,g} = \begin{cases} v_{j,i,g} & \text{if } \text{rand}_j[0,1] \leqslant Cr \text{ or } j = j_{\text{rand}} \\ x_{j,i,g} & \text{otherwise} \end{cases}$$

- Selección:

Si el vector $u_{i,g}$ tiene mejor valor de la función objetivo que su correspondiente vector objetivo $x_{i,g}$, sustituye el vector objetivo en la generación (g+1); si esto no ocurre, el vector objetivo permanece en la generación (g+1).

Esquema DE

Procedimiento Básico – Evolución Diferencial

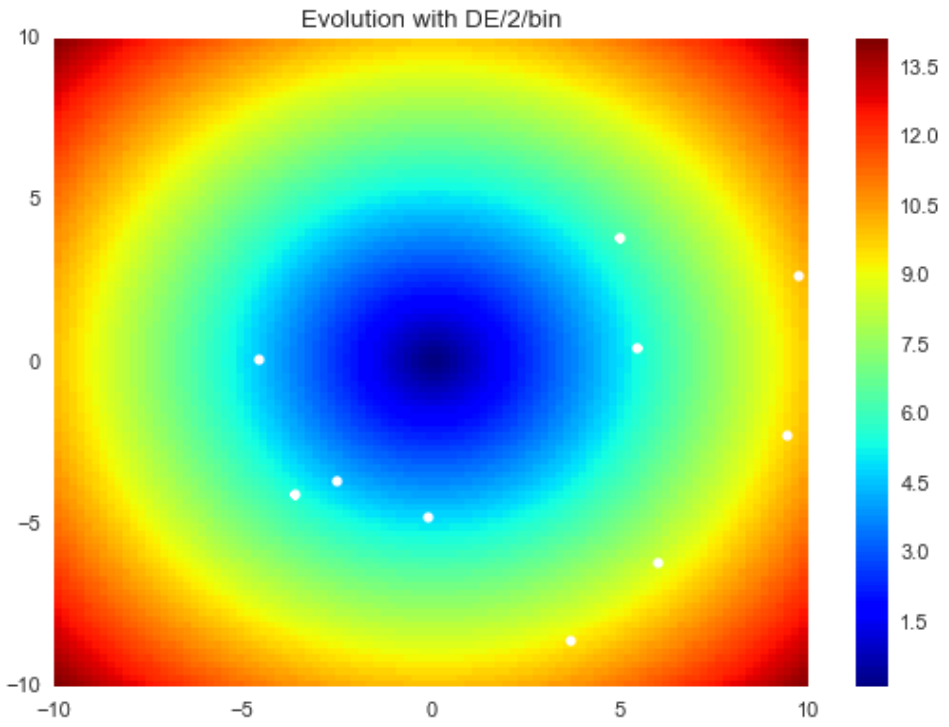
```
Procedure DE{
  t = 0;
  Initialize Pop(t);  /* of |Pop(t)| Individuals */
  Evaluate Pop(t);
  While (Not Done)
  {for i = 1 to |Pop(t)| do
       {parent1, parent2, parent3} = Select_3_Parents(Pop(t));
       thisGene = random_int(|Pop(t)|;
       for k = 1 to n do /* n genes per Individual */
          if (random < p) *P is crossover constant in [0,1]*
             Offspring_ik = parent1_ik + γ(parent2_ik – parent3_ik);
          else
             Offspring_ik = Individual_ik in Pop(t);
         end /* for k */
       Evaluate(Offspring_i);
      end /* for i */
    Pop(t+1) = {j | Offspring_j is_better_than Individual_j} ∪
               {k | Individual_k is_better_than Offspring_k};
    t = t + 1;}
```

CÓDIGO: http://www.icsi.berkeley.edu/~storn/code.html

Características de DE

- Método simple y efectivo.
- Aspectos importantes:
  - Población inicial, es muy importante.
- Desventajas:
  - Parámetro F fijo no es adecuado.
  - Valor correcto de CR depende de las dependencias entre variables:
    - Bastante separable: CR Bajo.
    - No separable: CR Alto.

Ejemplo de aplicación de DE



Trabajos de DE

R. Storn and K. V. Price, “Differential evolution-A simple and Efficient Heuristic for Global Optimization over Continuous Spaces,” *Journal of Global Optimization*, 11:341-359,1997.

K. V. Price, R. Storn, J. Lampinen, *Differential Evolution - A Practical Approach to Global Optimization*, Springer, Berlin, 2005.

U. K. Chakraborty, *Advances in Differential Evolution, Heidelberg, Germany: Springer-Verlag, 2008.*

Adaptando parámetros

- Distintos modelos adaptan parámetros
  - SaDE, JADE, SHADE, LSHADE.
- SaDE:
  - F según normal N(0.5, 0.2)
  - CR según normal N(CRmean, 0.1)
  - Adapta CRmean durante las ejecuciones.
- JADE
  - F también sigue N(Fmean, 0.3)
  - Adapta tanto Crmean como Fmean

Das, S., Maity, S., Qu, B.-Y., & Suganthan, P. N. (2011). Real-parameter evolutionary multimodal optimization — A survey of the state-of-the-art. Swarm and Evolutionary Computation, 1(2), 71–88. doi:10.1016/j.swevo.2011.05.005

Adaptando parámetros

- Distintos modelos adaptan parámetros
  - SaDE, JADE, SHADE, LSHADE.
- SHADE:
  - Aplica un almacén de soluciones anteriores.
  - No usa un único Crmean y Fmean, sino varios que adapta.
- L-SHADE
  - Tamaño Inicial más grande (más diversidad)
  - La población se reduce de forma lineal (los mejores)

Das, S., Maity, S., Qu, B.-Y., & Suganthan, P. N. (2011). Real-parameter evolutionary multimodal optimization — A survey of the state-of-the-art. Swarm and Evolutionary Computation, 1(2), 71–88. doi:10.1016/j.swevo.2011.05.005

Conclusiones del DE

Los algoritmos de Evolución Diferencial son uno de los campos más activos en el desarrollo de algoritmos evolutivos para la optimización de parámetros (optimización continua).

La hibridación de los algoritmos de Evolución Diferencial y los algoritmos de búsqueda local son una vía de trabajo con potenciales buenos resultados en el ámbito de la optimización continua.

Review reciente:

Swagatam Das *and Ponnuthurai Nagaratnam Suganthan*
Differential Evolution: A Survey of the State-of-the-Art,
IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION 15 (1): 4-31 (2011)

Optimización Inteligente (II)

1. Swarm Intelligence

2. Tipos de Optimización

3. *Frameworks* de Algoritmos Evolutivos

4. Caso de Estudio (Problemas continuos)

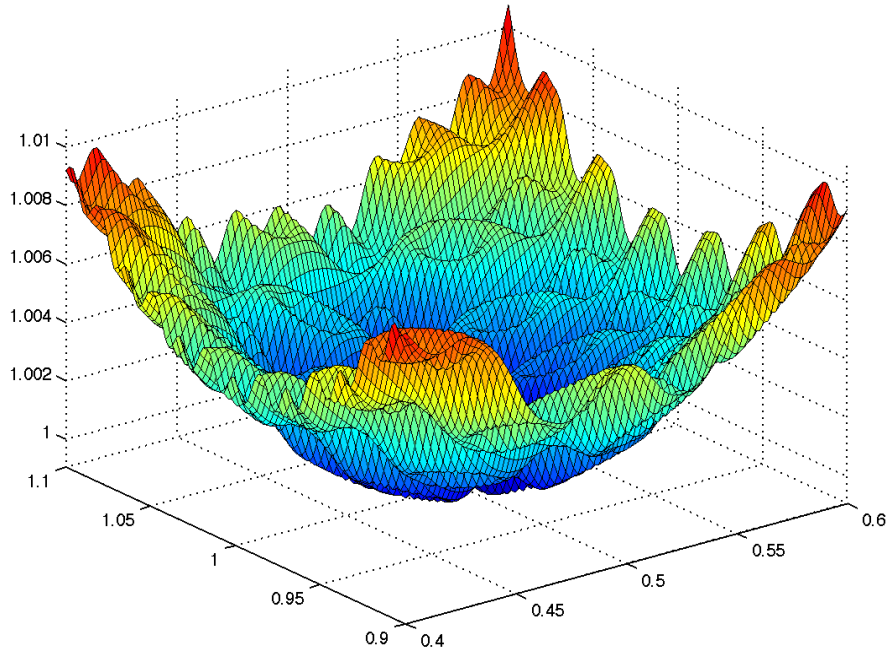
Optimización continua

- Optimizar un vector de variables reales, cada una con un rango.

Global Optima $f(x^*) \leq f(x) \quad \forall x \in Domain$

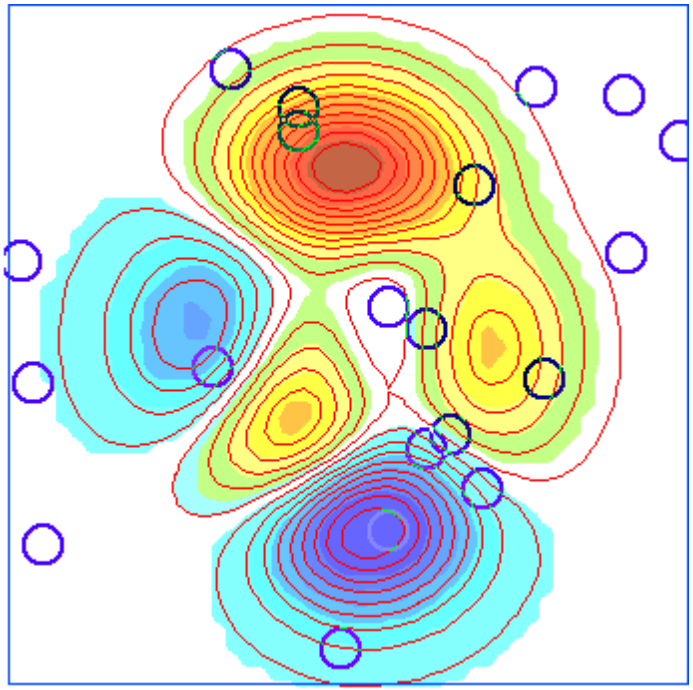
Real-parameter Optimization $Domain \subseteq \Re^D$,

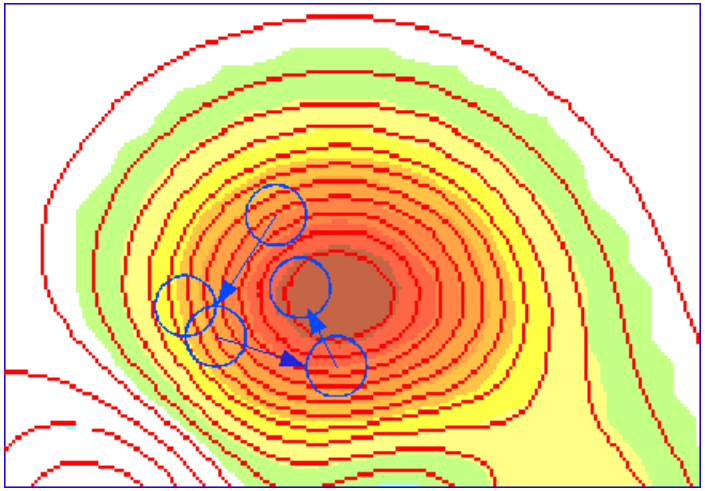
$x^* = [x_1, x_2, \cdots, x_D]$



Optimización continua

- La dificultad se incrementa con la dimensión.
- El espacio de búsqueda es complejo.
- Requiere un buen equilibrio entre exploración y explotación.

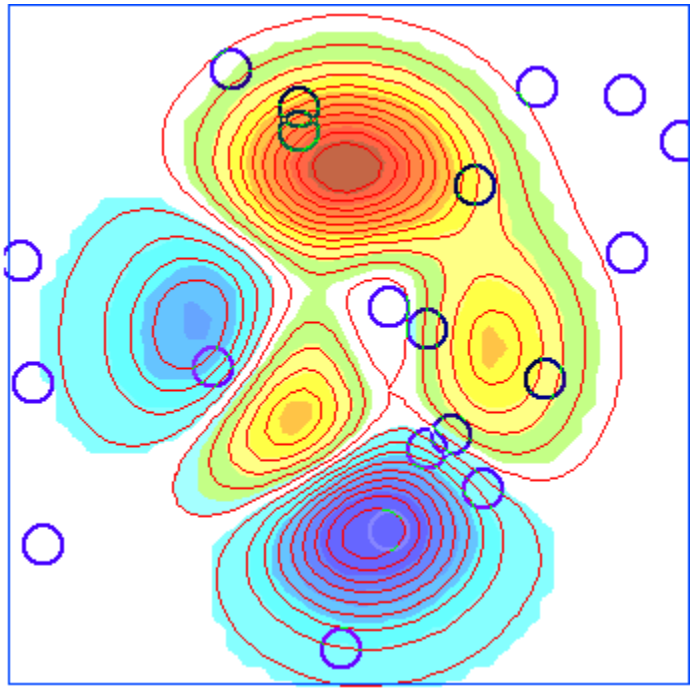




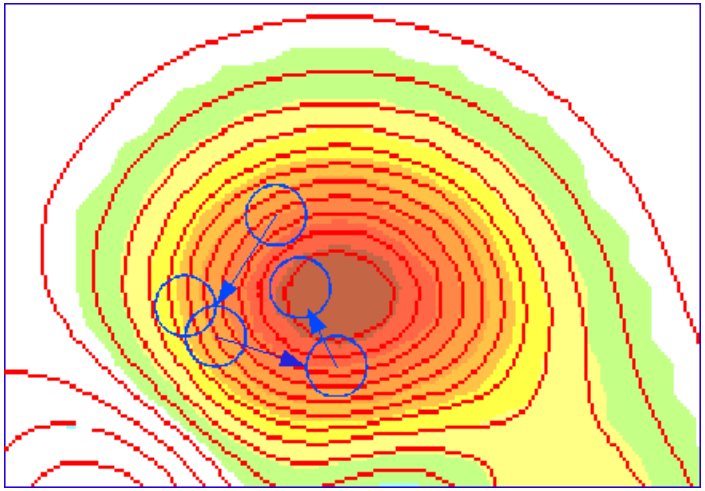
Optimización continua

- Muy usado el uso de Algoritmos Meméticos.
- Algoritmo Evolutivo para Explorar.

Algoritmo Evolutivo

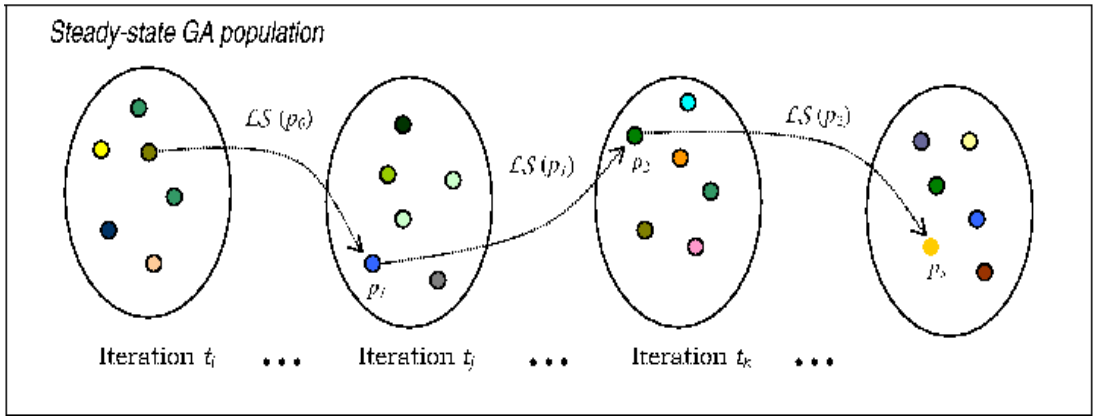


Búsqueda Local



Optimización continua

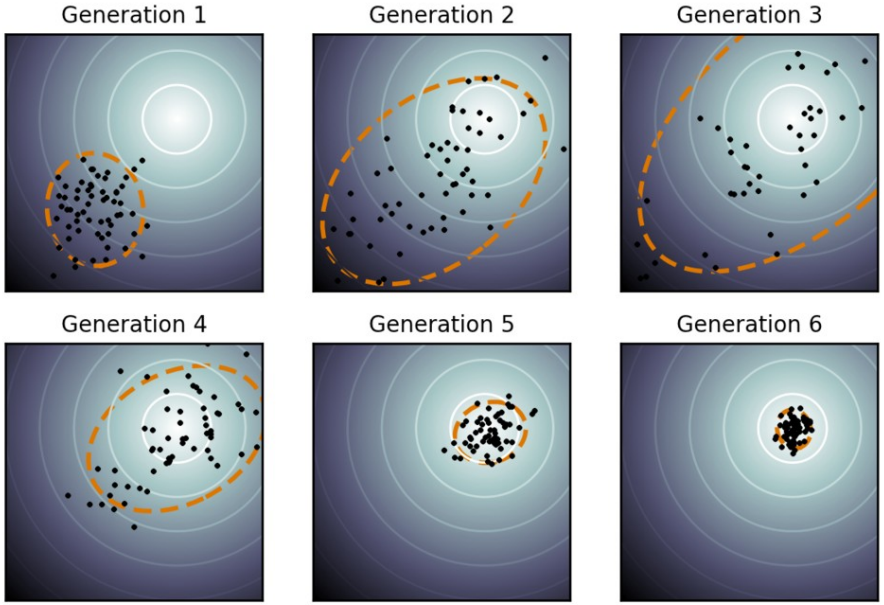
- ¿A quién aplicar la BL?
  - Mejor, que no haya mejorado.
- ¿Cúanto tiempo?
  - Ideal, continuar la búsqueda.
- ¿Atascado en Búsqueda Local?
  - Reinicio.



Algoritmos Recomendados

Poca dimensión:

C-CMAES



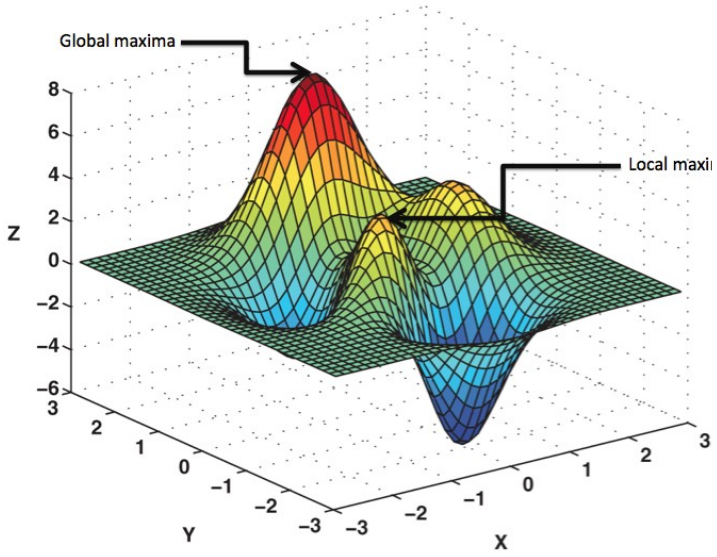
Mayor dimensión:

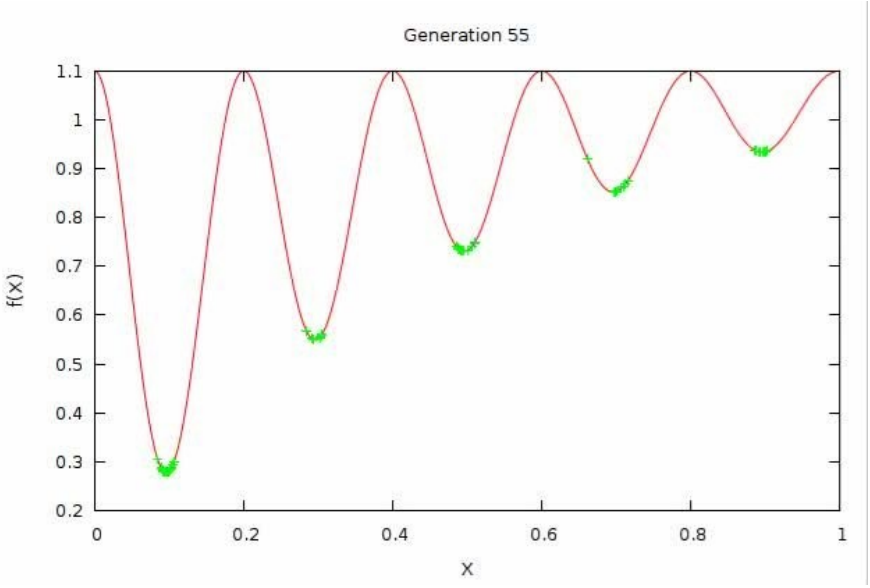
Versiones de DE

- DE clásico (pocas)
- Adaptativo:
  - SHADE
  - L-SHADE

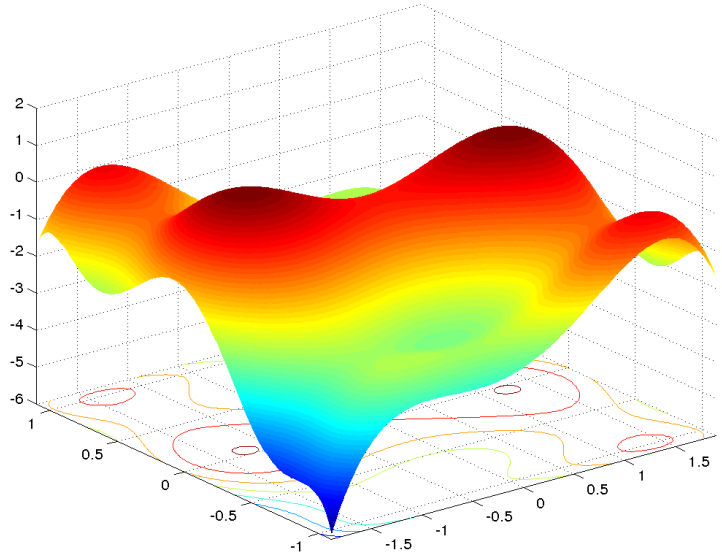
Optimización Multimodal

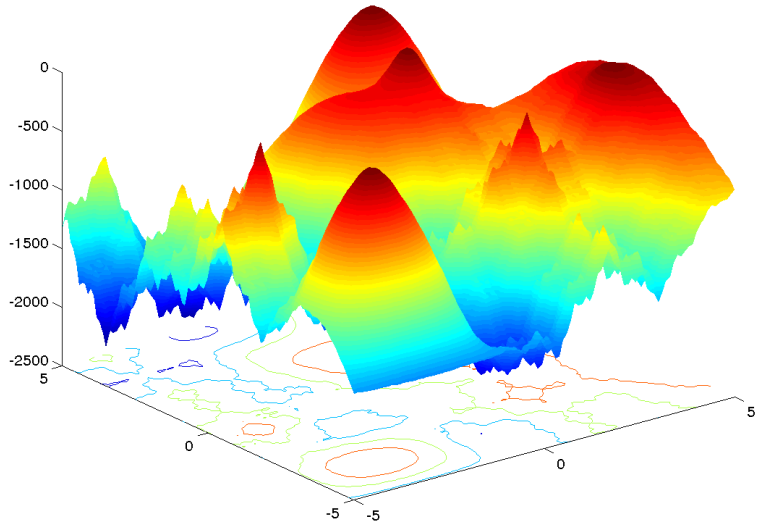
- Se desea obtener todos los óptimos, no uno solo.
- Evitan quedarse atascado prematuramente en óptimos.





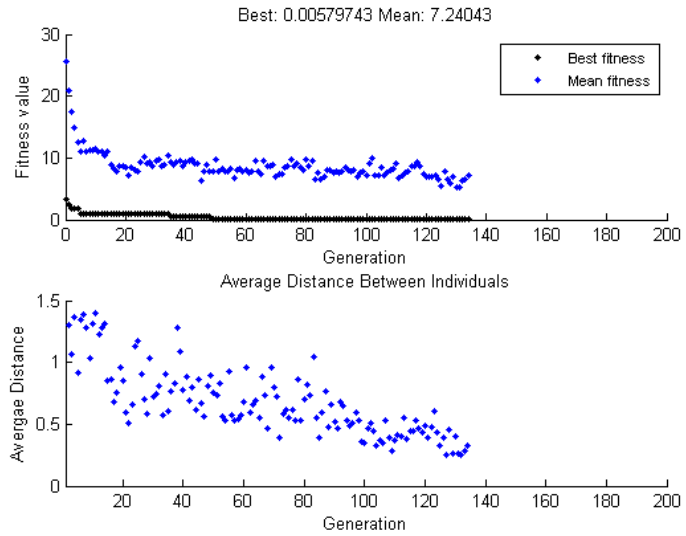
Funciones multimodales

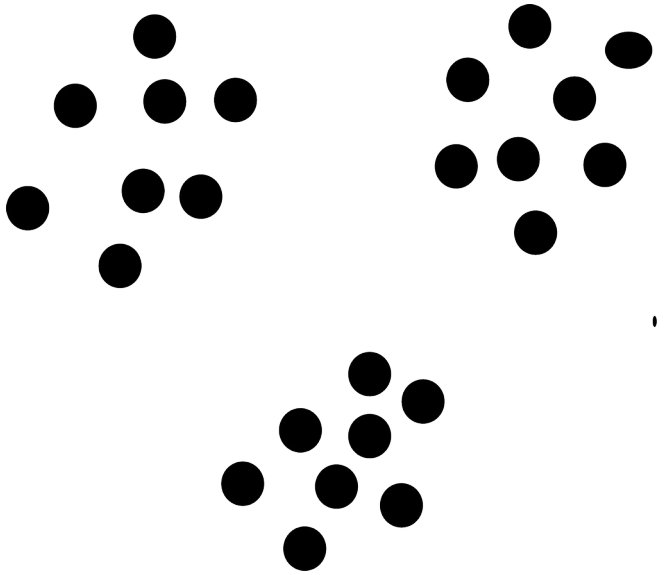




- Hay funciones con múltiples óptimos.
  - Locales o Reales.
- Muchos algoritmos no se comportan bien en esos casos.

Optimización Multimodal

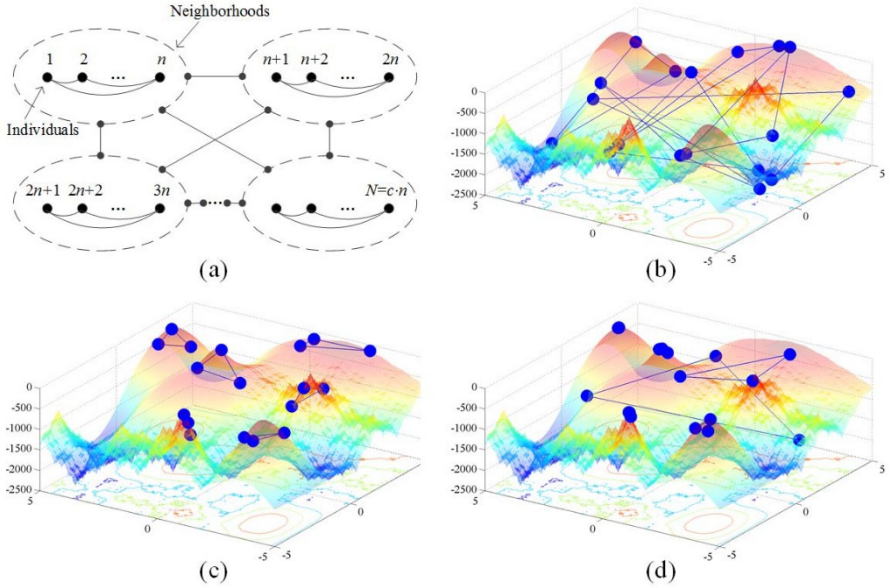




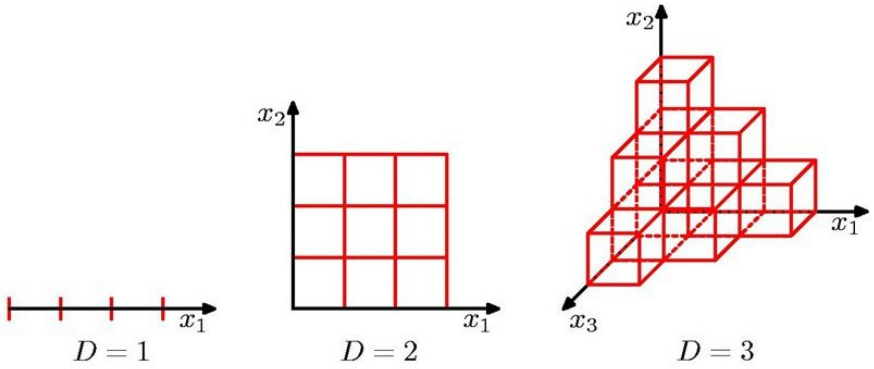
- Buscar métodos para separar las soluciones.
- Uso de subpoblaciones.
- Técnicas de *niching*:
  - *Sharing:* Penalizar soluciones cerca de otras.
  - *Clearing*: Eliminar soluciones demasiado cercanas.
  - *Crowding:* Reemplazar solución más cercana.

Algoritmos Recomendados

- Algoritmos clásicos mantienen mejores, no van bien, deben eliminarse de población.
- Aplican técnicas de nichos o **subpoblaciones**.
- Algoritmos específicos: NEA2 es el más conocido.



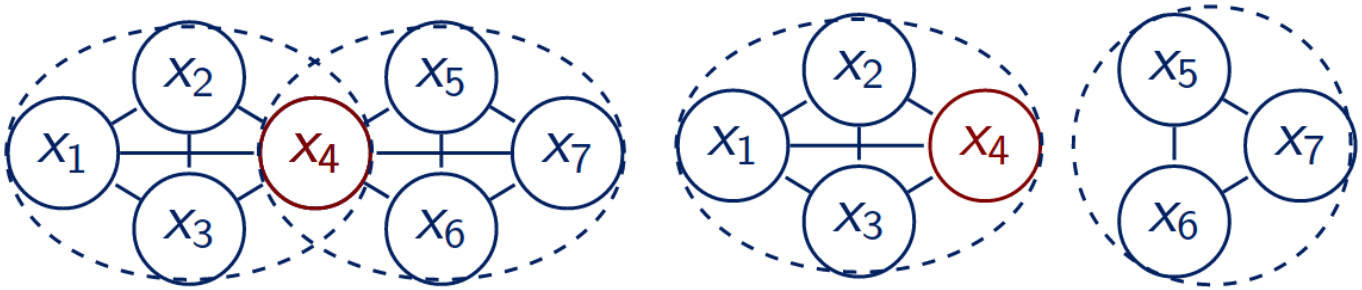
Alta dimensionalidad



- El espacio de búsqueda aumenta exponencialmente con la dimensionalidad.
- El número de evaluaciones (o tiempo) no puede aumentar tanto.
- Requiere algoritmos específicos.

Alta dimensionalidad

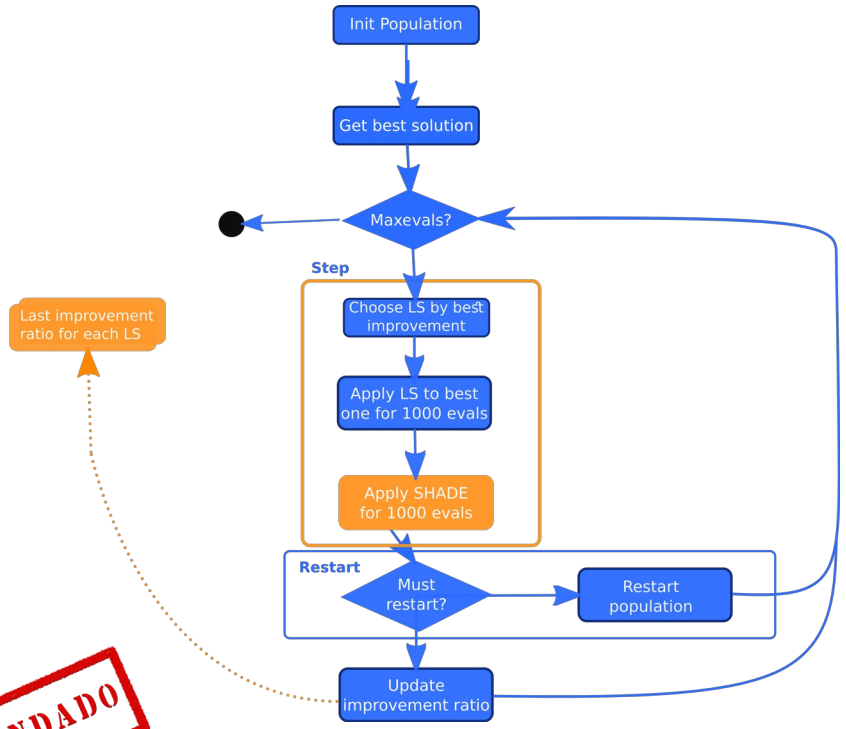
- Técnicas de descomposición



- Algoritmos
  - Differential Grouping.
  - Modelos recursivos.
- Ventajas y desventajas:
  - Mejora la exploración.
  - Puede consumir demasiado tiempo/evaluaciones.

Alta dimensionalidad

- MOS, algoritmo híbrido referencia durante años.
  - Adapta varios algoritmos.
- SHADEILS, actual estado del arte, ganador de la competición CEC'2018.
  - Dos métodos de BL.
  - Aplicación adaptativa.

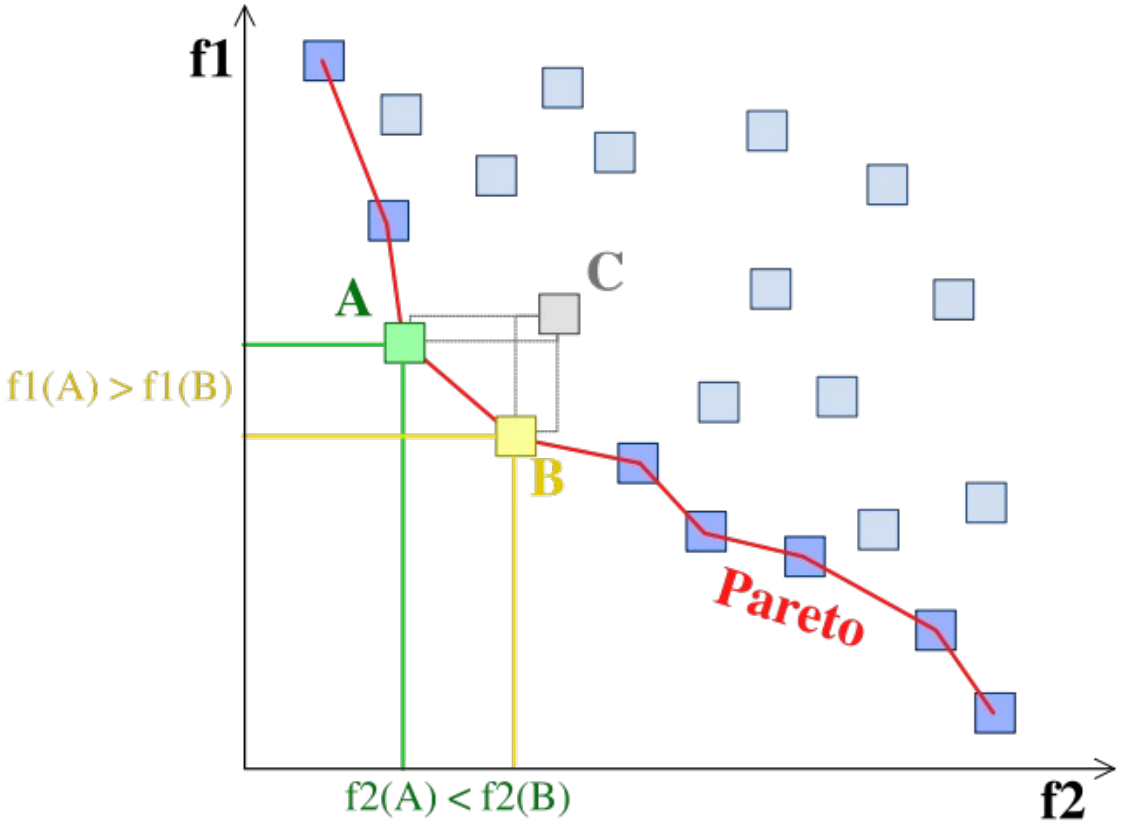


Problemas multiobjetivo

- A menudo interesa optimizar más de un objetivo a la vez.
- A menudo son contradirectorios.
- No siempre una solución es mejor que otra.

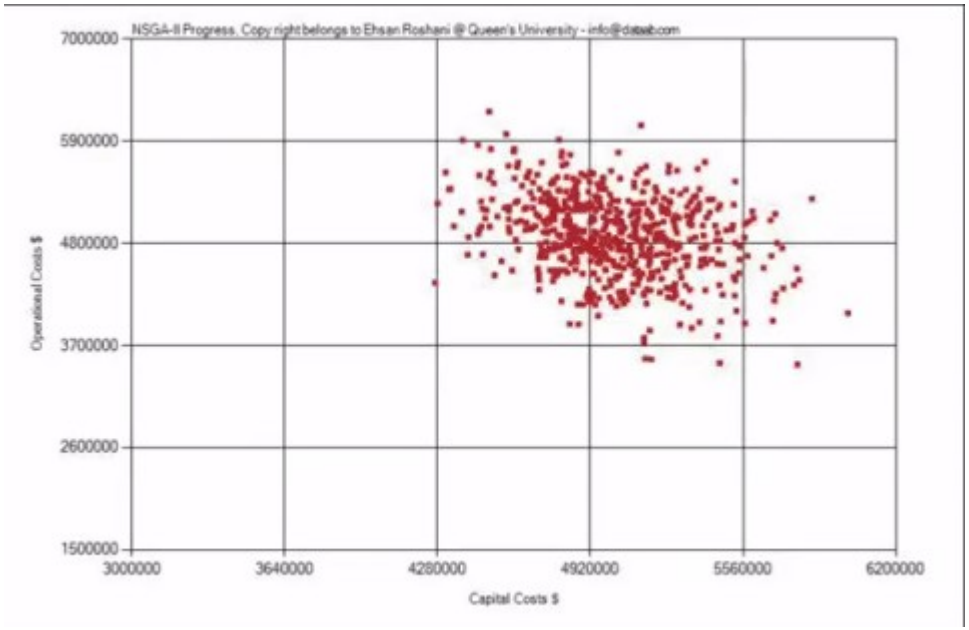
- Soluciones no dominadas:
- Si $f_1(x_i)$ mejor que $f_1(x_j)$ pero $f_2(x_i)$ peor que $f_2(x_j)$
  - Se dicen no dominadas.
- Los algoritmos buscan soluciones no dominadas.

Problemas multiobjetivo



Frente de Pareto

Problemas multiobjetivo

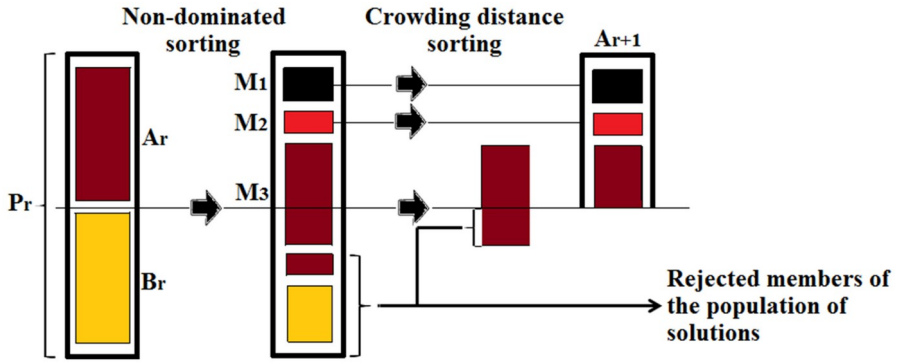


https://www.youtube.com/watch?v=sEEiGM9em8s

Algoritmos Recomendados

- Multi-objetivo: 2-3 objetivos

MOEA/D
NSGA-II



- Many-Objective: más de 3 objetivos.

NSGA-III

Optimización Inteligente (II)

1. Swarm Intelligence

2. Tipos de Optimización

3. *Frameworks* de Algoritmos Evolutivos

4. Caso de Estudio (Problemas continuos)

Software Disponible

- jMetal (Java) ó jMetalPy (Python)
  - Algoritmos Multi-objetivo.
  - Fácil de usar, problemas propios.
  - Se programa, permite gráficas.

- PlatEMO
  - Muchos algoritmos.
  - Implementado en Matlab.
  - Bien documentado.

Software Disponible

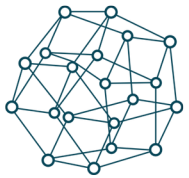
- ECJ, Java
  - Configurable con ficheros de texto.
  - Sólo requiere programar la función objetivo.
  -
- ParaDisEO, C++
  - Paralelismo.
  - Complejo de integrar un problema.
  - Gráficas.
- Inspyred, Python
  - Muchos algoritmos.
  - Fácil de usar.
- EO, C++
  - Completo.
  - Buen documentado.
  - Antigüo, problemas.

Software Disponible específico

- Mealpy, Python
  - Muchos Bio-inspirados.
  - Continuo crecimiento.
- NiaPy, Python
  - Nature-Inspired
  - Extension.

- PyaDE, Python
  - Algoritmos DE.
  - Modelos adaptativos.
- PySwarm, Python
  - Completo.
  - Buen documentado.
  - Gráficas

PYSWARMS
a research toolkit for Particle Swarm Optimization in Python



¿Preguntas?