Control y Optimización

Métodos heurísticos.
Algorítmos Genéticos, Particle Swarm, Simulated Annealing

Ignacio Delgado, Ignacio Gómez, José M. Perales, José M. Vega

Optimización Local y Global

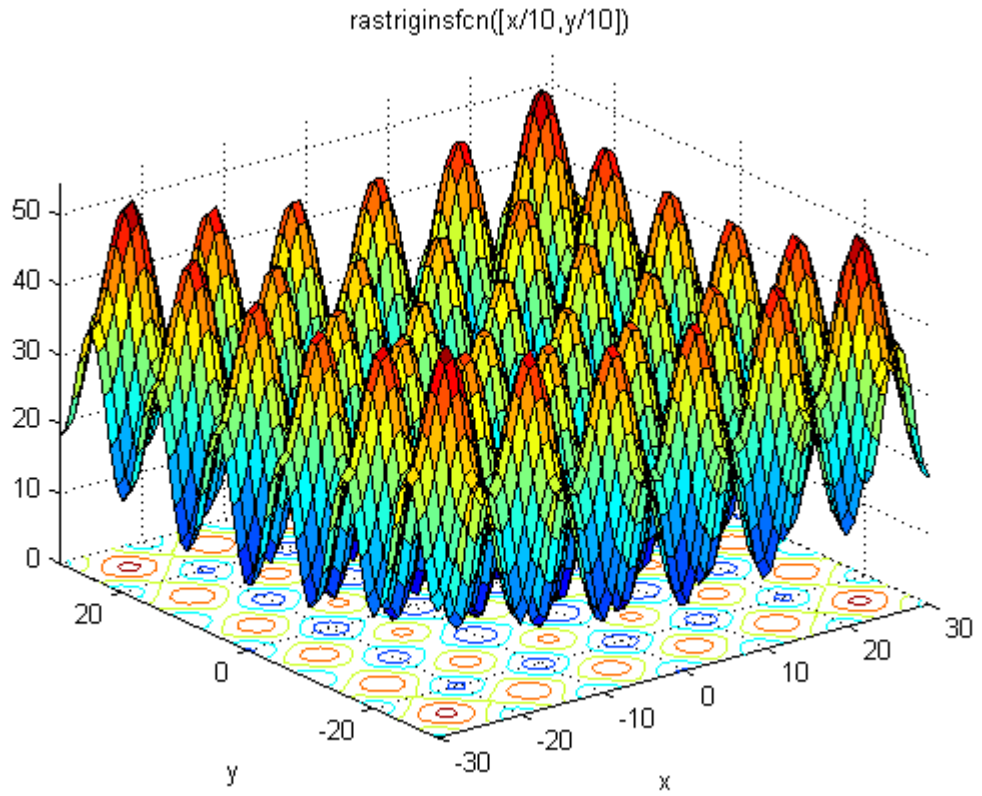
Los métodos anteriores (basados en la exploración en la dirección del gradiente), encuentran soluciones locales.

¿Qué ocurre si hay mas de un óptimo local?

¿Es factible encontrar el óptimo global (el óptimo de los óptimos?

Función ‘dificil’ de encontrar el óptimo global

$$Ras(x) = 20 + x_1^2 + x_2^2 - 10\left(\cos 2\pi x_1 + \cos 2\pi x_2\right)$$

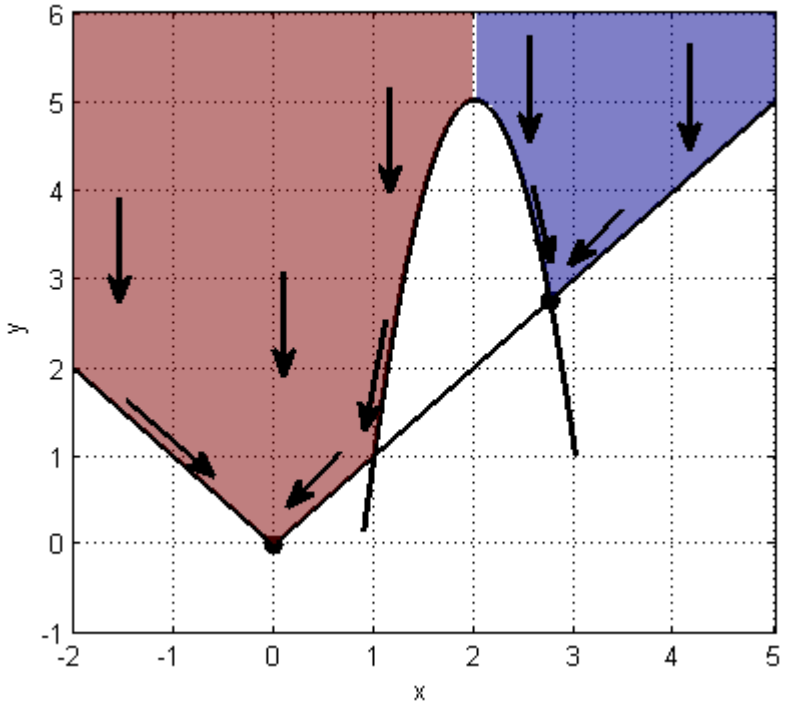


Comparación de resultados

Resultados	fminunc	patternsearch	ga	globalsearch
solucion	[19.9 29.9]	[19.9 -9.9]	[9.94 -0.01]	[0 0]
Función obj.	12.9	4.97	0.99	0
# Fevals	15	174	1040	2312

Dominios de atracción

¿Por qué depende el resultado del punto de partida?



Métodos Heurísticos

Un método heurístico es un método basado en la experiencia que puede utilizarse como ayuda para (posiblemente) encontrar un resultado (en este caso el valor del óptimo).

¿Por qué usar métodos heurísticos?

- Se pueden usar para resolver problemas de optimización combinatoria complejos (grandes, no lineales, no convexos -con varios mínimos locales-) que son difíciles de optimizar.
- A diferencia de los métodos basados en el gradiente en un espacio de diseño convexo, NO se garantiza que los métodos heurísticos encuentren el verdadero óptimo global, pero deberían encontrar muchas soluciones buenas (la respuesta del matemático frente a la del ingeniero)
- Los métodos heurísticos conviven bien con los óptimos locales sin quedarse atrapados mientras buscan el óptimo global.

Métodos Heurísticos (II)

¿Qué permiten, en general, los métodos heurísticos?

- 'Superar' los óptimos locales y no quedar atrapado en ellos.

- La optimización de sistemas en los que las variables de diseño no son solo continuas, sino también discretas, enteras o incluso booleanas

Estos métodos NO garantizan que se encuentre el óptimo global

Tipos de Métodos Heurísticos

Los métodos heurística incorporan con mucha frecuencia aleatoriedad.

Dos tipos de uso:

Uso para diseño preliminar:

- Primero debe encontrar una solución factible y luego mejorarla.

Uso para diseño incremental:

- Se parte de una solución factible y se intenta mejorar

Algunos Métodos Heurísticos:

- Algoritmos genéticos
- Recocido simulado (simulated annealing)
- Enjambre de partículas (particle swarm)
- Busqueda con tabú (Tabu Search)
- Fuegos artificiales (fireworks), ….

Algunos Métodos Heurísticos

Algoritmos genéticos (Holland, 1975)

Inspirada por la genética y la selección natural -maximo 'fitness'

Recocido simulado, Simulated Annealing (Kirkpatrick, 1983)

Inspirado por la mecánica estadística - minimización de la energía

Optimización de enjambre de partículas, Particle Swarm (Eberhart Kennedy, 1995)

Inspirado por el comportamiento social de enjambres de insectos o bandadas de pájaros – maximización de la "comida"

Todas estos métodos usan una combinación de aleatoriedad y de "reglas" heurísticas para guiar la búsqueda de máximos o mínimos globales.

Algoritmos Genéticos. Premisas

La selección natural es un principio de organización muy eficaz para optimizar poblaciones de individuos (y individuos)

• Si se puede imitar el proceso de la selección natural, se podría optimizar con éxito

• Un posible diseño de un sistema (representado por su vector de variables de diseño **u**) puede considerarse como un individuo que lucha por la supervivencia dentro de una población más grande.

• Solo los más aptos sobreviven. La 'adaptación' (fitness) se evalúa a través de la función objetivo F.

Algoritmos genéticos

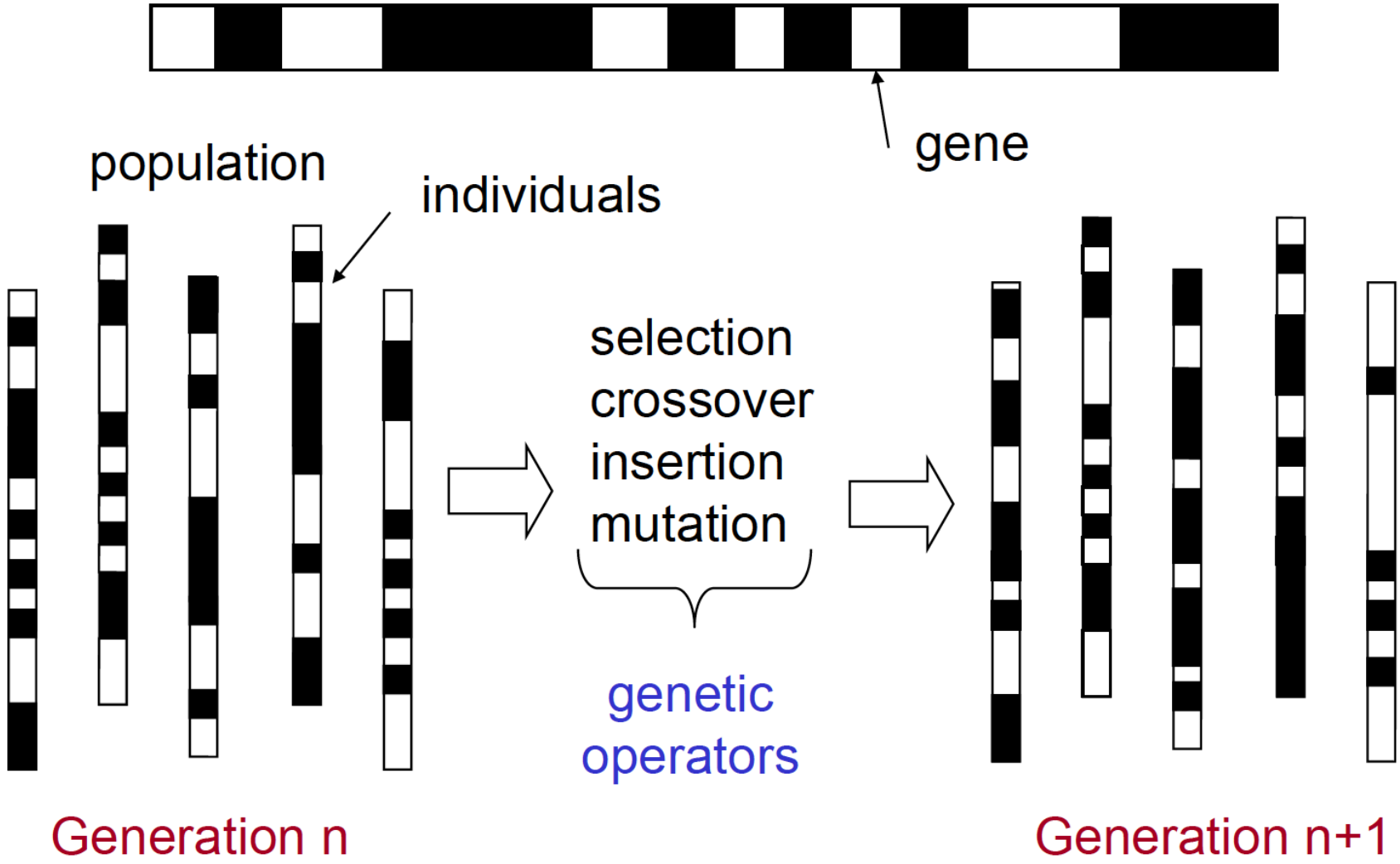
¿Qué se puede intentar cuando los métodos 'clásicos' fallan?: Algoritmos Genéticos, Particle Swarm, Simulated Annealing,…

- Algoritmos genéticos: Algoritmos inspirados en los principios de la evolución
- Se utilizan en problemas donde no se pueden encontrar soluciones con otros métodos o éstas no son satisfactorias
- Idea general: Se genera de forma aleatoria una población inicial de soluciones potenciales y se realiza un proceso iterativo que trasforma la población a través de:
  - Una evaluación (con la función de coste) de los individuos que forman la población
  - Una selección de las "mejores" y reproducción
  - Una recombinación de éstas formando una nueva población

Componentes de un algoritmo genético

- Una forma de hacer una representación genética (cromosomas) de las posibles soluciones del problema
- Una forma de crear una población inicial de soluciones
- Una función de evaluación capaz de medir la adaptación (fitness) de cualquier solución
- Un conjunto de operadores genéticos como reglas de transición probabilísticas para guiar la búsqueda
- El valor de unos parámetros de entrada que el algoritmo genético utilizará para guiar su evolución

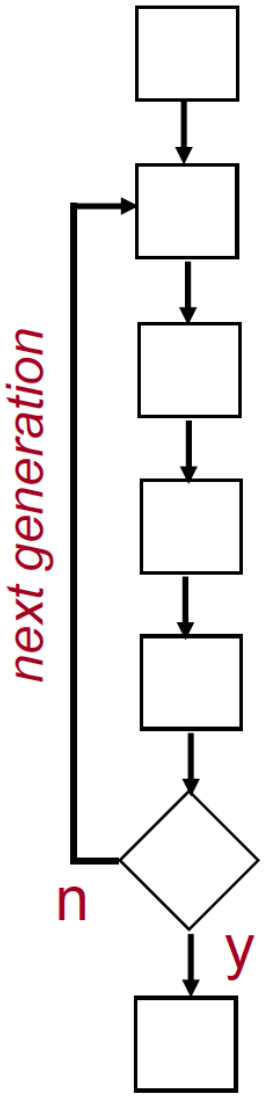
Algoritmos genéticos. Terminologia



Pseudocodigo de un Algoritmo genético

```
begin
  t=0;
  inicializar P(t);
  evaluar P(t);
  while (no se cumpla la condición de finalizar)
  do
     t=t+1;
     reproducir P(t-1) en P(t);
     recombinar P(t);
     evaluar P(t);
  end while;
end.
```

Pseudocodigo de un Algoritmo genético



Initialize Population (initialization)

Select individual for mating (selection)

Mate individuals and produce children (crossover)

Mutate children (mutation)

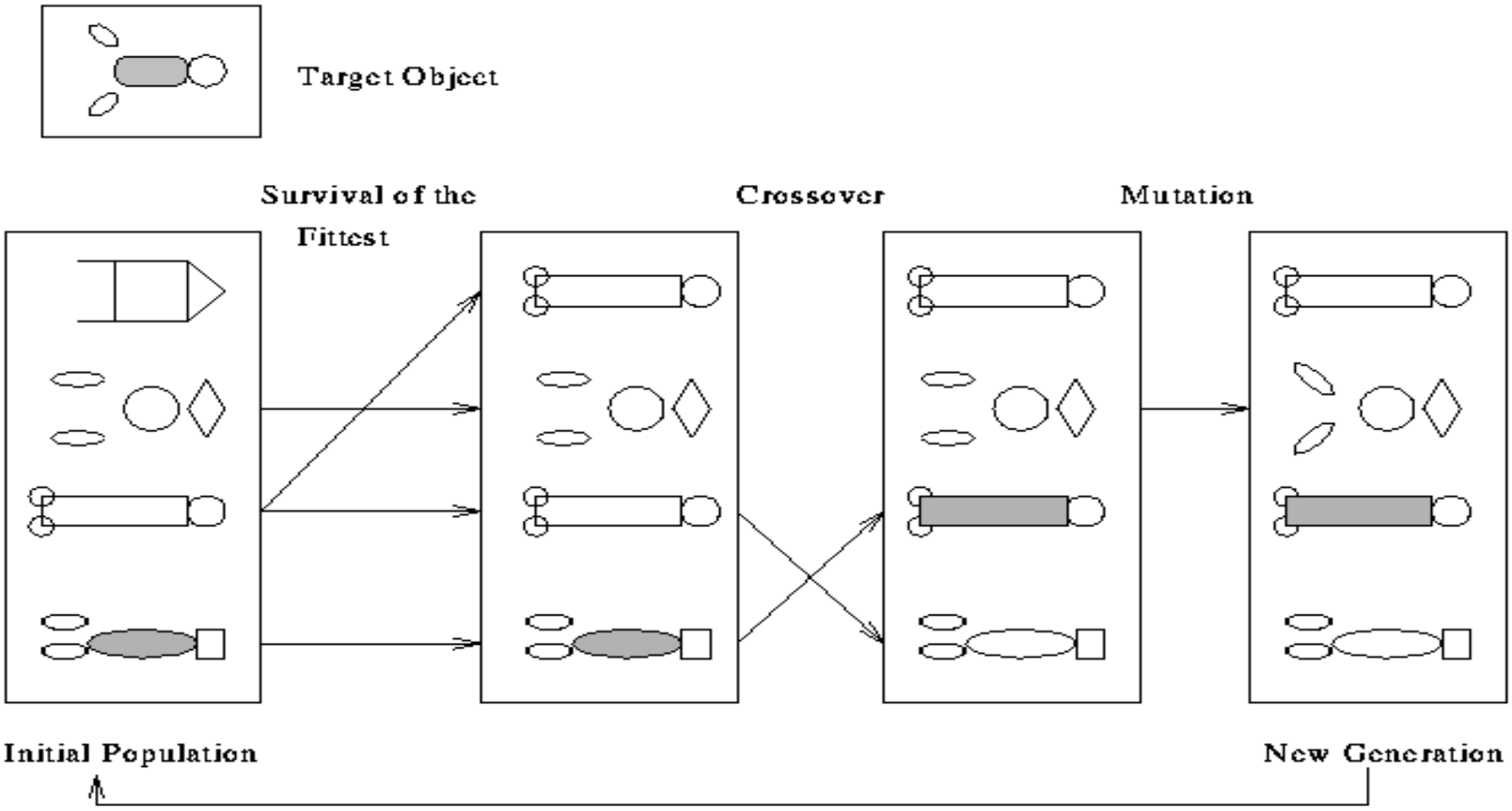
Insert children into population (insertion)

Are stopping criteria satisfied ?

Finish

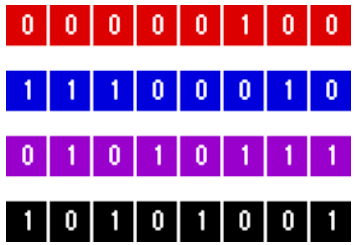
Ref: Goldberg (1989)

Esquema básico



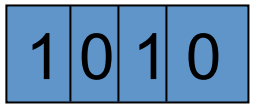
Representación (Codificación)

Cadenas binarias (generalmente) de longitud determinada por el número de variables existentes en una solución y el número de bits necesarios para representarlas; cromosomas o estructuras



Ejemplo: maximizar $f(x)=x^2$ (0 - 31)

Representación en binario



Los cromosomas están compuestos por genes; el valor de un gen se denomina alelo y a su posición locus

Representación

- Cada iteración será una generación; $c_i^t = (b_{i1}^t ... b_{il}^t)$ representa el cromosoma c_i de la generación t, b será un gen o un elemento del vocabulario elegido.

- Podemos representar un individuo X_i^t en una generación determinada t, como la terna:

$$X_i^t = (c_i^t, x_i^t, f_i^t)$$

- x_i^t es la decodificación del cromosoma (fenotipo) y f_i^t es la adecuación de la solución

Obtención de la población inicial

- Conjunto de m individuos, $P(t)= \{X_i^t,....X_i^t\}$
- La inicialización de un individuo X_i^0 consiste en asignar un valor aleatorio a cada uno de los genes b_{ij}^0, con la decodificación obtenemos su fenotipo x_i^0 y f_i^0 lo obtendremos a través de la función de evaluación.

$m = 4$

0 1 1 0 1
1 1 0 0 0
0 1 0 0 0
1 0 0 1 1

Función de evaluación

- **Objetivo:** medir la adaptación (fitness) de una solución en una generación t
- La función de evaluación se corresponde con la función objetivo del problema
- Dado un cromosoma c_i^t, y su fenotipo x_i^t podemos obtener su adecuación f_i^t como:
  - $f_i^t = \text{eval}(c_i^t) = f(x_i^t)$

Función de evaluación

$f_i^t = \text{eval}(c_i^t) = f(x_i^t)$ en este caso $f(x) = x^2$

Codificación c_i	Decodificación x_i	Fitness f_i
01101	13	169
11000	24	576
01000	8	64
10011	19	361
		Σ=1170

Función Objetivo (Fitness)

Elegir la función objetivo correcta es muy importante, pero también bastante difícil

Los Algorítmos Genéticos NO admiten "restricciones explícitas"

Las restricciones se pueden manejar de diferentes maneras:

- a través de la función de fitness (funciones barrera)
- a través del operador de selección (rechazo de los individuos que violan la restricción)
- implícitamente a través de la representación/codificación (que sea capaz de implementar sólo individuos válidos)

➢ Elegir la función de fitness adecuada es un paso importante en el diseño de un algoritmo genético

Operadores Genéticos: Reproducción

Reproducción: Incluye un algoritmo de selección y un algoritmo de muestreo

- El algoritmo de selección asigna una probabilidad de selección a cada cromosoma
- El algoritmo de muestreo produce copias de los cromosomas de la generación $t-1$ a la generación t

Los cromosomas con mayor probabilidad de selección se reproducirán un número de veces mayor y tendrán mayor repercusión en las siguientes generaciones.

Operadores Genéticos: Selección y Muestreo

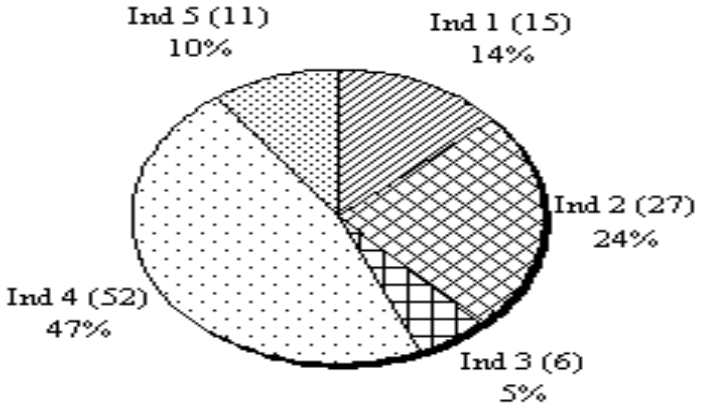
c_i	x_i	Fitness	% del total	
01101	13	169	14.4	→ 1 copia
11000	24	576	49.2	→ 2 copias
01000	8	64	5.5	→ 0 copias
10011	19	361	30.9	→ 1 copia
		Σ=1170	Σ=100	

Operadores Genéticos: esquemas de selección y muestreo

- **Basado en el rango:**
  - Se mantiene el porcentaje de la población.
  - Los M peores se substituyen por la descendencia de $N-M$ mejores
- **Rueda de ruleta:**
  - Los cromosomas de la generación actual en una cantidad proporcional a su "bondad"
- **Selección de torneo**:
  - Se escoge aleatoriamente un número T de individuos, gana el que mejor se adapta, se repite hasta obtener el número de individuos deseados

Operadores Genéticos: Rueda de ruleta

Population	*Fitness*
Individual 1	15
Individual 2	27
Individual 3	6
Individual 4	52
Individual 5	11



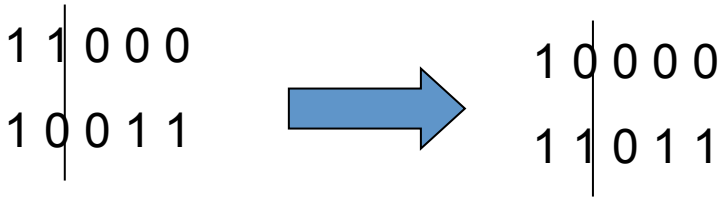
Random ly generated number = 21

Individual 2 is selected

Operadores Genéticos: Cruce

Cruce:

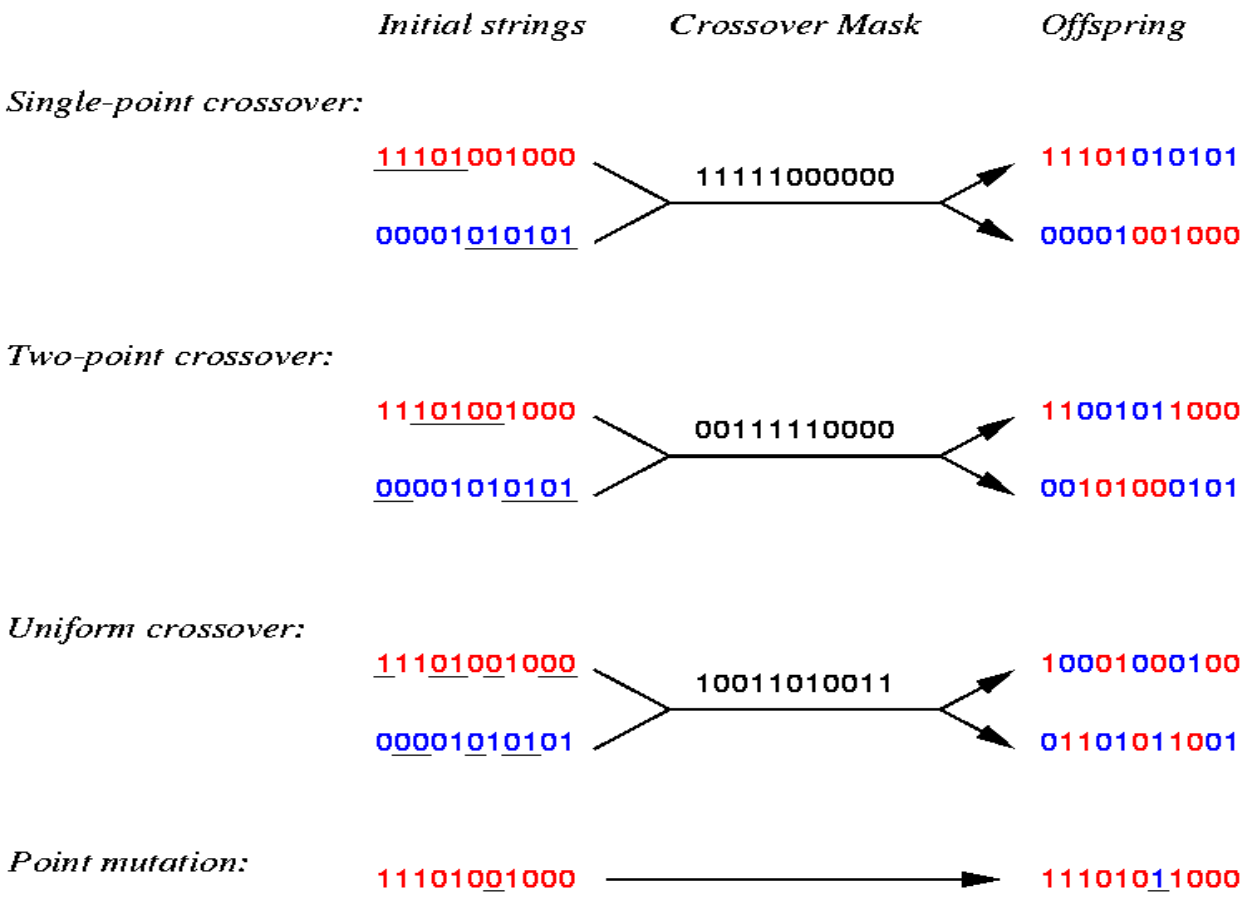
- Dependiendo de una probabilidad inicial, probabilidad de cruce seleccionamos de forma aleatoria los cromosomas que van a participar en el apareamiento
- A continuación aplicamos alguna técnica de cruce, por ejemplo el *cruce simple*



Operadores Genéticos: Cruce

- Cruce de *n* puntos:
  - los cromosomas se cortan por *n* puntos aleatorios y se intercambia el material genético
- Cruce uniforme:
  - cada gen se obtiene de la madre o del padre de forma aleatoria

Operadores Genéticos: Cruce



Operadores Genéticos: Mutación

- **Mutación**:
  - Su objetivo es producir diversidad en la población
  - Teniendo en cuenta una probabilidad, probabilidad de mutación, y de forma aleatoria se altera un bit o gen de un cromosoma

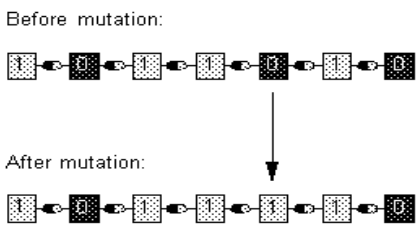


Figure 5.4: Mutation.

- Una vez aplicados los operadores, se evalúa de nuevo la población

Algoritmos Genéticos (y Matlab)

- **NO hace falta codificarlos. Ya están implementados en diferentes librerias**

  Por ejemplo en Matlab (funciones `ga` y `gamultiobj`)

- **Sólo hay que modificar algunos parámetros (o dejar los valores por defecto), por ejemplo** :
  - Tamaño de la población
  - Número de generaciones
  - Probabilidad de cruce
  - Probabilidad de mutación

Diferencia entre los Alg. Genéticos y otros métodos

- Los AG buscan una población de puntos en paralelo, no solo un punto
- Los AG usan reglas de transición probabilísticas, no deterministas
- Los AG trabajan sobre una codificación del conjunto de variables de diseño más que en las variables mismas
- Los AG no requieren información sobre el valor del gradiente u otros conocimientos auxiliares; solo se necesita la función objetivo y sus valores correspondientes influyen en la búsqueda (aleatoria).

Ventajas de los Algoritmos Genéticos

- Funcionan bien en problemas mixtos discretos/continuos
- Requieren poca información sobre el problema
- No se requieren gradientes
- Son fáciles de entender y configurar e implementar
- Son muy robustos
- Son algoritmos estocásticos, es decir, explotan la aleatoriedad
- Se pueden paralelizar fácilmente

Desventajas de los Algoritmos Genéticos

- La codificación de los AG sigue siendo un arte y requiere algo de experiencia (pero ya están incluidos en librerías)
- La convergencia puede ser dependiente de algunos parámetros de ajuste: tasa de mutación, cruzamiento, tamaño de la población, …
- El diseño de la función de fitness puede ser complicado
- El tratamiento de las limitaciones puede ser engorroso
- Pueden ser computacionalmente costosos
- No hay criterios claros de finalización
- No hay un 'conocimiento' del verdadero óptimo global

El problema de la mochila con Algoritmos Genéticos

- N objetos para meter en una mochila
- Cada objeto i, tiene un peso p_i, y si se mete en la mochila produce un beneficio b_i
- **Objetivo:** Llenar la mochila obteniendo el máximo beneficio:
- Maximizar $F = \sum_{1 \le i \le n} b_i x_i$ con la restricción $\sum_{1 \le i \le n} p_i x_i \le C$

$$x_i \in \{0,1\}, b_i > 0, p_i > 0$$

El problema de la mochila con AG

- **Representación**

  $X=(x_1, x_2,...x_n)$ x_i es (0,1)

  Puede haber individuos que no cumplen las restricciones (que sean demasiado pesados para la mochila)

- **Función de evaluación**:

$$F = \begin{cases} C - \sum_{1 \le i \le n} b_i x_i & \text{si} \quad \sum_{1 \le i \le n} b_i x_i > C \\ \sum_{1 \le i \le n} b_i x_i & \text{si} \quad \sum_{1 \le i \le n} b_i x_i \le C \end{cases}$$

El Problema de la mochila con AG

- Con la función de evaluación eliminamos las soluciones no factibles
- Se generan individuos de forma aleatoria
- Para el operador de cruce, se puede usar el cruce simple
- Se puede seguir el esquema general

Fundamentos de los Algoritmos Genéticos. Esquema

Esquema es una subcadena dentro de un individuo

Cada esquema es una posible razón por la cual el individuo tiene la adaptación que tiene

Cada cadena de longitud K tiene 2^K esquemas incluidos

Por ejemplo, la cadena 101 tiene los siguientes esquemas posibles: 101, 10*, 1*1, *01, 1**, *0*, **1, ***

donde * es un comodín y puede ser llenado con cualquier símbolo

101 es el esquema más específico, y *** el más general

Cada individuo prueba 2^K hipótesis (esquemas) a la vez

Fundamentos de los Algoritmos Genéticos

- **Esquema (Schema)**: patrón de similitud que describe un subconjunto de cadenas con similitudes en ciertas posiciones.
- Aumentamos el vocabulario (que era 0,1) con el símbolo *, en las posiciones en las que aparece este símbolo puede haber cualquier elemento del alfabeto inicial.
- **Órden** de un esquema $O(E)$: número de posiciones fijas en él.

Fundamentos de los Algoritmos Genéticos

- **Longitud** de un esquema $L(E)$: distancia entre la primera y la última posición definida en el esquema
- Cada ristra pertenece a todas las regiones (esquemas) en las cuales aparece cualquiera de sus bits
- El número de esquemas procesados eficazmente por un algoritmo genético que maneje una población de n individuos es del orden de n^3. ***Paralelismo implícito***

Fundamentos de los Algoritmos Genéticos

Aplicación del operador selección:

La probabilidad de seleccionar un cromosoma perteneciente a un esquema E, viene dada por el cociente entre la adecuación media de los representantes de un esquema y la adecuación media de la población en un instante t:

$$m(E,t+1) = m(E,t) \cdot f_{pro}(E)/f_{pro}$$

donde $m(E,t)$ es el número de representantes del esquema E en la generación t

Fundamentos de los Algoritmos Genéticos

Aplicación del operador cruce:

- Si A=B no se destruye ningún esquema
- Si el orden del esquema es cero, no será destruido nunca
- Si la longitud del esquema es uno la probabilidad de que sea destruido es $1/(q-1)$ donde q es la longitud de la cadena
- Si la longitud del esquema es dos la probabilidad es $2/m-1$. En general $L(E)/(q-1)$
- Teniendo en cuenta además la probabilidad de aplicar el cruce, la probabilidad es $P_c\ L(E)/(q-1)$

Fundamentos de los Algoritmos Genéticos

Aplicación del operador mutación:

Si P_m es la probabilidad de mutación, $1-P_m$ es la probabilidad de que un gen sobreviva y $O(E)P_m$ la probabilidad de que sobrevivan todos los de un cromosoma

Si consideramos un esquema por encima de la media de adaptación (fitness) e, por el efecto del operador reproducción, en cada generación los esquemas que tienen un valor por encima de la media reciben un incremento exponencial de sus representantes

$$m(E,t+1) = m(E,0)(1+e)^t$$

Fundamentos de los Algoritmos Genéticos

Si consideramos también el efecto del operador cruce y mutación:

$$m(E,t+1) >= m(E,t) \cdot (f_{pro}(E)/f_{pro}) \cdot [1 - P_c L(E)/(q-1) - O(E)P_m]$$

Teorema del esquema (Holland):

Los esquemas de longitud corta, orden bajo y que llevan a una adecuación por encima de la media de la generación reciben un incremento exponencial en las subsiguientes generaciones cuando se les aplica una iteración de un algoritmo genético

Origen del Simulated Annealing

El simulated annealing es una técnica heurística que matemáticamente trata de simular el enfriamiento de un conjunto de átomos a un estado de energía mínima.

- Origen: aplicación del campo de Mecánica Estadística en el campo de la Optimización Combinatoria (1983)
- Dibuja una analogía entre el enfriamiento de un material (búsqueda del estado de energía mínimo) y la resolución de un problema de optimización.
- Articulo original/Introducción del concepto

  Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P., "Optimization by Simulated Annealing," *Science*, 220, 4598, 13 May 1983, pp. 671-680.

Simulated Annealing

El simulated annealing sirve para resolver problemas de optimización sin restricciones y con variables con límites inferiores y superiores

El método modela el proceso físico del calentamiento inicial de un material y una bajada lenta posterior la temperatura para disminuir los defectos, lo que minimiza la energía del sistema.

Simulated Annealing

- En cada iteración del algoritmo, se genera aleatoriamente un nuevo punto. La distancia al nuevo punto desde el punto actual, o la extensión de la búsqueda, se basa en una distribución de probabilidad con una escala proporcional a la 'temperatura'.
- El algoritmo acepta todos los puntos nuevos que disminuyen el objetivo, pero también, con una cierta probabilidad, puntos que lo incrementan. Al aceptar puntos que lo incrementan, se evita quedar atrapado en mínimos locales, y se puede explorar otras regiones para obtener más soluciones posibles.
- Se selecciona un programa de 'recocido' para disminuir sistemáticamente la temperatura a medida que avanza el algoritmo. A medida que la temperatura disminuye, el algoritmo va reduciendo el alcance de su búsqueda para converger finalmente a un mínimo.

Algoritmo usado en el Simulated Annealing

Nomeclatura:

$\mathbf{x}$ = Vector de variables de diseño (o R)

E = Energía del sistema (es decir, valor de la función objetivo)

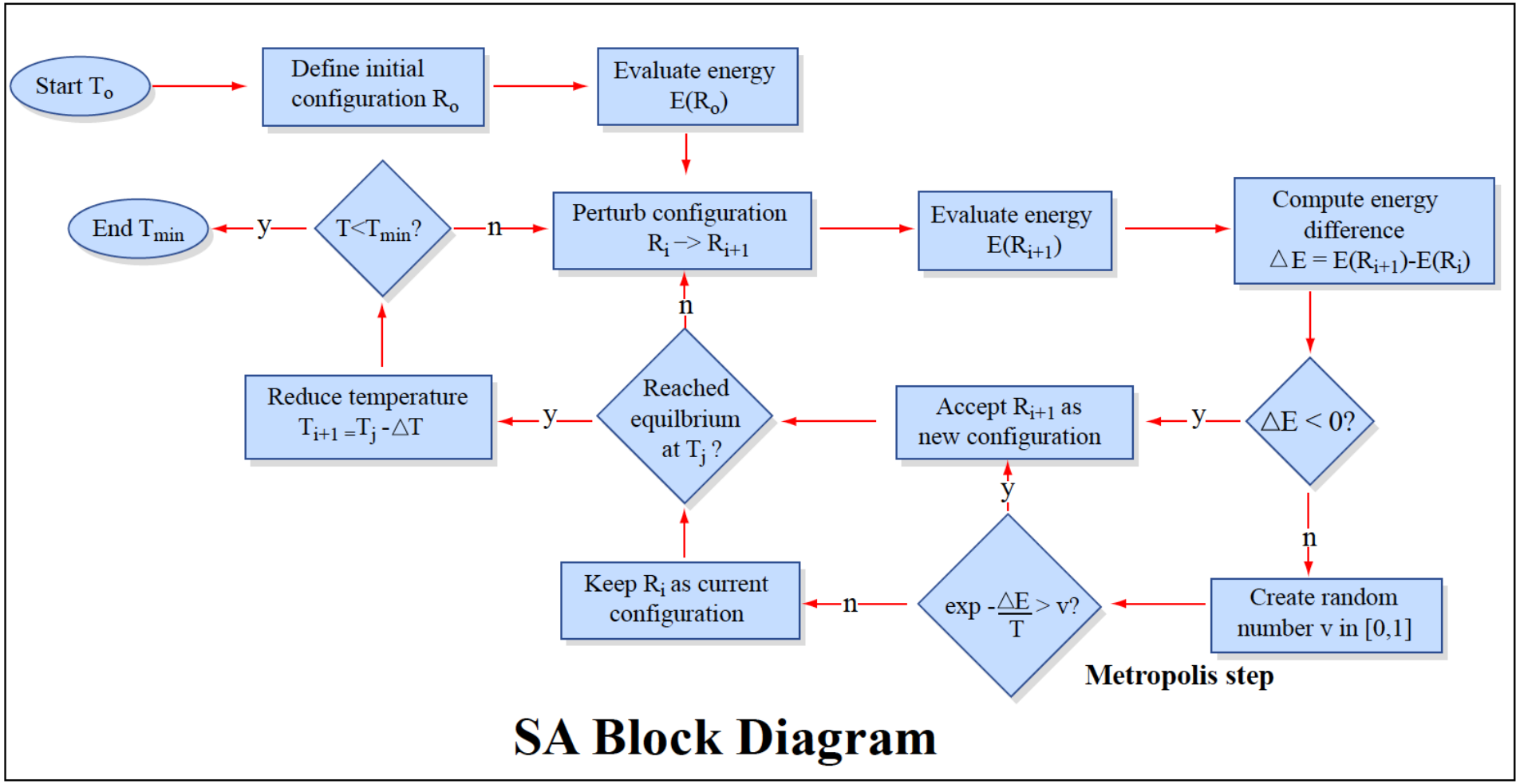
T = Temperatura del sistema

Δ = Diferencia en la energía del sistema entre dos vectores de diseño

El algoritmo de recocido simulado

1. Se elije un $\mathbf{x}_i$ aleatorio, se selecciona la temperatura inicial del sistema y se especifica el cronograma de enfriamiento (es decir, recocido)
2. Se evalúa $E(\mathbf{x}_i)$
3. Se perturba $\mathbf{x}_i$ para obtener un Vector de variables de diseño próximo, $\mathbf{x}_{i+1}$
4. Se evalúa $E(\mathbf{x}_{i+1})$
5. Si $E(\mathbf{x}_{i+1})<E(\mathbf{x}_i)$, $\mathbf{x}_{i+1}$ es el nuevo valor
6. Si $E(\mathbf{x}_{i+1})>E(\mathbf{x}_i)$, entonces se acepta $\mathbf{x}_{i+1}$ como nuevo punto con una probabilidad $e^{(-\Delta/T)}$ donde $\Delta=E(\mathbf{x}_{i+1})-E(\mathbf{x}_i)$
7. Se reduce la temperatura del sistema de acuerdo con el 'programa de enfriamiento'
8. Se repite desde el paso 2

Diagrama de bloques del algoritmo de Simulated Annealing



Aspectos a tener en cuenta en el algoritmo de Simulated Annealing

- Una descripción concisa de una configuración del problema (Vector de variables de diseño).
- Un generador aleatorio de puntos próximos a una configuración (Vecindarios). Este generador puede incluir reglas para generar solo configuraciones válidas (Función de perturbación).
- Una función objetivo cuantitativa (análogo de la energía del sistema).
- Un programa de recocido de las temperaturas y/o la duración de los tiempos (iteraciones) para los cuales se dejará evolucionar el algoritmo (el sistema).

Enjambre de partículas (Particle Swarm)

Es un método heurístico de optimización inspirado en la inteligencia colectiva de enjambres de poblaciones biológicas.

Por ejemplo: Bandadas de Pájaros o Enjambres de Insectos

Conceptos detrás del Particle Swarm

¿Cómo hacen un gran número pájaros aves para mantener la 'formación' aun cuando a menudo cambian de dirección, se dispersan y reagrupan?

- Procesos locales "descentralizados".
- Manipulación de distancias entre individuos (para mantener el ritmo y evitar la colisión).

¿Hay ventajas en el comportamiento como enjambre/bandada para un individuo?

- Puede beneficiarse de los descubrimientos y la experiencia previa de otros miembros del enjambre en la búsqueda de alimentos, evitando a los depredadores, ajustándose al medio ambiente, es decir, el intercambio de información produce una ventaja evolutiva.

Algoritmo de Particle Swarm

Descripción de las partículas (cada partícula tiene tres características):

- Posición, $\mathbf{x}_k^i$, (la *i*-esima partícula en el momento *k*)
- Velocidad, $\mathbf{v}_k^i$, (similar a la dirección de búsqueda, se utilizará para actualizar la posición)
- Valor de la función objetivo (o Fitness), $f\left(\mathbf{x}_k^i\right)$, u objetivo (determina qué partícula tiene el mejor valor en el enjambre y también determina la mejor posición de cada partícula a lo largo del tiempo

Algoritmo de Particle Swarm

Enjambre inicial

- No hay criterios completamente establecidos para determinar el tamaño de enjambre, normalmente de 10 a 60.
- Las partículas se distribuyen aleatoriamente en el espacio de diseño. $\mathbf{x}_0^i = \mathbf{x}_{\min} + \text{rand} \cdot \left(\mathbf{x}_{\max} - \mathbf{x}_{\min} \right)$

  donde $\mathbf{x}_{\min}$ y $\mathbf{x}_{\max}$ son los valores de los límites inferiores y de los límites superiores de las variables respectivamente
- Se evalúa la aptitud (función objetivo) de cada partícula
  - la mejor posición conocida de la partícula $\left(\mathbf{p}^i \text{, que aqui coincide con } \mathbf{x}_0^i \right)$
  - la mejor posición en el enjambre actual (influencia del enjambre, $\mathbf{p}_0^g$)
- La velocidad inicial se genera aleatoriamente

$$\mathbf{v}_0^i = \frac{\mathbf{x}_{\min} + \text{rand} \cdot \left(\mathbf{x}_{\max} - \mathbf{x}_{\min} \right)}{\Delta t} = \frac{\text{posición}}{\text{tiempo}}$$

Algoritmo de Particle Swarm

Actualización de la velocidad

- Proporciona direcciones de búsqueda
- Incluye parámetros determinísticos y probabilísticos.
- Combina el efecto del movimiento en el momento, la memoria de la partícula, y la influencia del enjambre

$$\mathbf{v}_{k+1}^{i} = w\mathbf{v}_{k}^{i} + c_1 \text{rand} \cdot \frac{\left(\mathbf{p}^{i} - \mathbf{x}_{k}^{i}\right)}{\Delta t} + c_2 \text{rand} \cdot \frac{\left(\mathbf{p}_{k}^{q} - \mathbf{x}_{k}^{i}\right)}{\Delta t}$$

w = factor de inercia (valores entre 0,4 y 1,4)

c_1 = auto-confianza (valores entre 1,5 y 2)

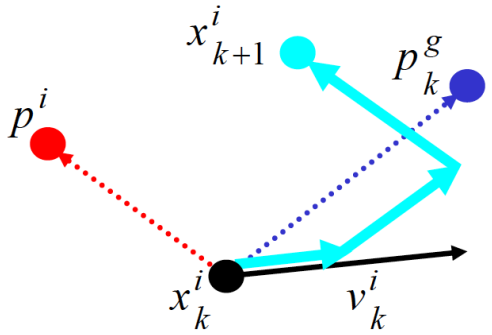
c_2 = confianza en el enjambre (valores entre 2 y 2,5)

Algoritmo de Particle Swarm

Actualización de la posición

La posición se actualiza por vector de velocidad

$$\mathbf{x}_{k+1}^{i} = \mathbf{x}_{k}^{i} + \mathbf{v}_{k+1}^{i}\Delta t$$



Criterio de parada

El cambio máximo (a lo largo de un número determinado de movimientos, S) en el punto de mejor valor de la función objetivo es menor que la tolerancia especificada

$$\left| f\left(\mathbf{p}_{k}^{q}\right) - f\left(\mathbf{p}_{k-q}^{q}\right) \right| \le \varepsilon \quad , \quad q = 1,2,3,\ldots,S$$