Certificando las soluciones dadas por algoritmos evolutivos

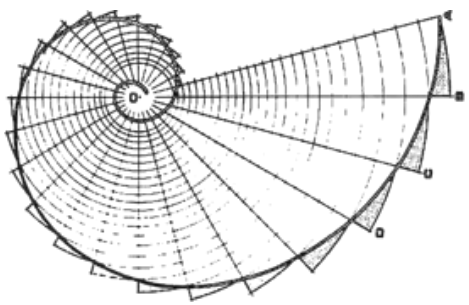
Cruz E. Borges
cruz.borges@deusto.es

Jose L. Montaña
montanjl@unican.es

Luis M. Pardo
luis.m.pardo@gmail.com

DeustoTech Energy

UC UNIVERSIDAD DE CANTABRIA



CONGRESO DE JÓVENES INVESTIGADORES

Real Sociedad Matemática Española

Universidad de Murcia, del 7 al 11 de Septiembre de 2015

Objetivo

Objetivo de la charla

- Dar un ejemplo de certificación de soluciones de algoritmos evolutivos
- Mostrar cómo usar lo anterior para estudiar la complejidad de un algoritmo evolutivo

Índice

Aviso

¡VOY A SIMPLIFICAR MUCHO EL DISCURSO!

«Antes simple que incomprensible»

Pedro - www.eltamiz.com

Aviso

¡VOY A SIMPLIFICAR MUCHO EL DISCURSO!

«Antes simple que incomprensible»

Pedro - www.eltamiz.com

Índice

Definición [controvertida]

Algoritmos que replican la forma en que la naturaleza se enfrenta a problema de optimización. Principalmente la selección natural y la respuesta de organismos vivos a distintos desafíos

Pseudocódigo

- Se genera una población inicial
- Esta se actualiza [mientras se cumpla un criterio] usando varios operadores:

Selección:	Selecciona individuos de la población
Evolución:	Genera nuevos individuos
Calificación:	Evalúa la calidad de los nuevos individuos
Reemplazo:	Selecciona los individuos que pasarán a la siguiente población

Definición [controvertida]

Algoritmos que replican la forma en que la naturaleza se enfrenta a problema de optimización. Principalmente la selección natural y la respuesta de organismos vivos a distintos desafíos

Pseudocódigo

- Se genera una población inicial
- Esta se actualiza [mientras se cumpla un criterio] usando varios operadores:

Selección:	Selecciona individuos de la población
Evolución:	Genera nuevos individuos
Calificación:	Evalúa la calidad de los nuevos individuos
Reemplazo:	Selecciona los individuos que pasarán a la siguiente población

Algoritmos Evolutivos

Ventajas y Desventajas

Ventajas

- Relativamente genéricos y con muy pocas restricciones
- Ofrecen poblaciones de soluciones no respuestas únicas
- Son intrínsecamente paralelos
- No parten de ideas preconcebidas

Desventajas

- Algoritmos intrínsecamente probabilistas
- Hasta el momento, su comportamiento es desconocido
- Tienen una cantidad muy grande de hiperparámetros libres
- En general, no es posible certificar soluciones

Algoritmos Evolutivos

Ventajas y Desventajas

Ventajas

- Relativamente genéricos y con muy pocas restricciones
- Ofrecen poblaciones de soluciones no respuestas únicas
- Son intrínsecamente paralelos
- No parten de ideas preconcebidas

Desventajas

- Algoritmos intrínsecamente probabilistas
- Hasta el momento, su comportamiento es desconocido
- Tienen una cantidad muy grande de hiperparámetros libres
- En general, no es posible certificar soluciones

Ejemplos

- Problemas de control, inteligencia artificial, reconocimiento de patrones, regresiones simbólicas, resolución simbólica de ecuaciones en derivadas parciales, resolución de sistemas de ecuaciones no lineales, …
- Diseño de estructuras como: alas de un avión, salas de concierto, antenas de telecomunicaciones, grúas, …
- Diseño de fármacos, compuesto químicos o materiales
- Reconocimiento de patologías en test clínicos de diversos tipos
- Predicciones sobre mercados financieros

Índice

Componentes

Estructura de Datos: [Codifica un candidato a solución]

$$\mathbb{A} := A_1 \times \cdots \times A_n \times \mathbb{N}^m \times \mathbb{R}^l$$

Operador de Calificación: [Especifica cuanto de bueno es el candidato a solución]

$$f : \mathbb{A} \longrightarrow \mathbb{R}$$

Operador de Evolución: [Modifica un candidato a solución]

$$e : \mathbb{A}^i \longrightarrow \mathbb{A}^o$$

Condición de parada: [Para que sea un algoritmo]

Número de iteraciones $<$ G

¿Cuál es el tamaño de la entrada de un algoritmo evolutivo?

Componentes

Estructura de Datos: [Codifica un candidato a solución]

$$\mathbb{A} := A_1 \times \cdots \times A_n \times \mathbb{N}^m \times \mathbb{R}^l$$

Operador de Calificación: [Especifica cuanto de bueno es el candidato a solución]

$$f : \mathbb{A} \longrightarrow \mathbb{R}$$

Operador de Evolución: [Modifica un candidato a solución]

$$e : \mathbb{A}^i \longrightarrow \mathbb{A}^o$$

Condición de parada: [Para que sea un algoritmo]

$$\text{Número de iteraciones} < G$$

¿Cuál es el tamaño de la entrada de un algoritmo evolutivo?

Análisis de la complejidad de un algoritmo evolutivo

Número de pasos y espacio utilizado

Espacio utilizado

$$\left|\mathbb{A}^{P}\right| + \mathcal{O}_e(f) + \varepsilon$$

donde P es el tamaño de la población y $\mathcal{O}_e(f)$ el espacio que necesita el operador de calificación

Número de pasos

$$G \times P \times (\mathcal{O}_t(f) + \varepsilon)$$

donde G es el número de iteraciones, P es el tamaño de la población y

$\mathcal{O}_t(f)$ el número de instrucciones que ejecuta el operador de calificación

¿Cómo escoger G y P?

Análisis de la complejidad de un algoritmo evolutivo

Número de pasos y espacio utilizado

Espacio utilizado

$$\left|\mathbb{A}^{P}\right| + \mathcal{O}_{e}(f) + \varepsilon$$

donde P es el tamaño de la población y $\mathcal{O}_{e}(f)$ el espacio que necesita el operador de calificación

Número de pasos

$$G \times P \times (\mathcal{O}_{t}(f) + \varepsilon)$$

donde G es el número de iteraciones, P es el tamaño de la población y

$\mathcal{O}_{t}(f)$ el número de instrucciones que ejecuta el operador de calificación

¿Cómo escoger G y P?

Análisis de la complejidad de un algoritmo evolutivo

Número de pasos y espacio utilizado

Espacio utilizado

$$\left|\mathbb{A}^{P}\right| + \mathcal{O}_{e}(f) + \varepsilon$$

donde P es el tamaño de la población y $\mathcal{O}_{e}(f)$ el espacio que necesita el operador de calificación

Número de pasos

$$G \times P \times (\mathcal{O}_{t}(f) + \varepsilon)$$

donde G es el número de iteraciones, P es el tamaño de la población y

$\mathcal{O}_{t}(f)$ el número de instrucciones que ejecuta el operador de calificación

¿Cómo escoger G y P?

Análisis de la complejidad de un algoritmo evolutivo

Complejidad en media

Análisis

1. Suponemos que tenemos una función $\mathcal{I} : \mathbb{A} \longrightarrow \{0, 1\}$ tal que nos indica si un elemento es una solución
2. Definimos una distribución de probabilidad ν en $\mathbb{A}$
3. Calculamos como *evoluciona* ν, es decir $C_j := E_\nu[\mathcal{I} \circ e^j]$
4. Rezas para que C_j sea creciente en j
5. Definimos G como el j tal que $C_j \geq \frac{2}{3}$
6. Si G es polinomial, ¡obtienes un algoritmo $\mathcal{BPP}$!

Problemas

Análisis de la complejidad de un algoritmo evolutivo

Complejidad en media

Análisis

1. Suponemos que tenemos una función $\mathcal{I} : \mathbb{A} \longrightarrow \{0, 1\}$ tal que nos indica si un elemento es una solución
2. Definimos una distribución de probabilidad ν en $\mathbb{A}$
3. Calculamos como *evoluciona* ν, es decir $C_j := E_\nu[\mathcal{I} \circ e^j]$
4. Rezas para que C_j sea creciente en j
5. Definimos G como el j tal que $C_j \geq \frac{2}{3}$
6. Si G es polinomial, ¡obtienes un algoritmo $\mathcal{BPP}$!

Problemas

Análisis de la complejidad de un algoritmo evolutivo

Complejidad en media

Análisis

1. Suponemos que tenemos una función $\mathcal{I} : \mathbb{A} \longrightarrow \{0, 1\}$ tal que nos indica si un elemento es una solución
2. Definimos una distribución de probabilidad ν en $\mathbb{A}$
3. Calculamos como *evoluciona* ν, es decir $C_j := E_\nu[\mathcal{I} \circ e^j]$
4. Rezas para que C_j sea creciente en j
5. Definimos G como el j tal que $C_j \geq \frac{2}{3}$
6. Si G es polinomial, ¡obtienes un algoritmo $\mathcal{BPP}$!

Problemas

Análisis de la complejidad de un algoritmo evolutivo

Complejidad en media

Análisis

1. Suponemos que tenemos una función $\mathcal{I} : \mathbb{A} \longrightarrow \{0, 1\}$ tal que nos indica si un elemento es una solución
2. Definimos una distribución de probabilidad ν en $\mathbb{A}$
3. Calculamos como *evoluciona* ν, es decir $C_j := \mathrm{E}_\nu[\mathcal{I} \circ e^j]$
4. Rezas para que C_j sea creciente en j
5. Definimos G como el j tal que $C_j \geq \frac{2}{3}$
6. Si G es polinomial, ¡obtienes un algoritmo $\mathcal{BPP}$!

Problemas

Análisis de la complejidad de un algoritmo evolutivo

Complejidad en media

Análisis

1. Suponemos que tenemos una función $\mathcal{I} : \mathbb{A} \longrightarrow \{0, 1\}$ tal que nos indica si un elemento es una solución
2. Definimos una distribución de probabilidad ν en $\mathbb{A}$
3. Calculamos como *evoluciona* ν, es decir $C_j := E_\nu[\mathcal{I} \circ e^j]$
4. Rezas para que C_j sea creciente en j
5. Definimos G como el j tal que $C_j \geq \frac{2}{3}$
6. Si G es polinomial, ¡obtienes un algoritmo $\mathcal{BPP}$!

Problemas

Análisis

1. Suponemos que tenemos una función $\mathcal{I} : \mathbb{A} \longrightarrow \{0, 1\}$ tal que nos indica si un elemento es una solución
2. Definimos una distribución de probabilidad ν en $\mathbb{A}$
3. Calculamos como *evoluciona* ν, es decir $C_j := E_\nu[\mathcal{I} \circ e^j]$
4. Rezas para que C_j sea creciente en j
5. Definimos G como el j tal que $C_j \geq \frac{2}{3}$
6. Si G es polinomial, ¡obtienes un algoritmo $\mathcal{BPP}$!

Problemas

Análisis de la complejidad de un algoritmo evolutivo

Complejidad en media

Análisis

1. Suponemos que tenemos una función $\mathcal{I} : \mathbb{A} \longrightarrow \{0, 1\}$ tal que nos indica si un elemento es una solución
2. Definimos una distribución de probabilidad ν en $\mathbb{A}$
3. Calculamos como *evoluciona* ν, es decir $C_j := \mathrm{E}_\nu[\mathcal{I} \circ e^j]$
4. Rezas para que C_j sea creciente en j
5. Definimos G como el j tal que $C_j \geq \frac{2}{3}$
6. Si G es polinomial, ¡obtienes un algoritmo $\mathcal{BPP}$!

Problemas

- ¿Cómo definimos $\mathcal{I}$?
- ¡Mucho ánimo calculando C_j!

Análisis de la complejidad de un algoritmo evolutivo

Complejidad en media

Análisis

1. Suponemos que tenemos una función $\mathcal{I} : \mathbb{A} \longrightarrow \{0, 1\}$ tal que nos indica si un elemento es una solución
2. Definimos una distribución de probabilidad ν en $\mathbb{A}$
3. Calculamos como *evoluciona* ν, es decir $C_j := \mathbb{E}_\nu[\mathcal{I} \circ e^j]$
4. Rezas para que C_j sea creciente en j
5. Definimos G como el j tal que $C_j \geq \frac{2}{3}$
6. Si G es polinomial, ¡obtienes un algoritmo $\mathcal{BPP}$!

Problemas

- ¿Cómo definimos $\mathcal{I}$?
- ¡Mucho ánimo calculando C_j!

Índice

El problema xvii de Smale

Problema xvii de Smale (Smale 2000)

¿Existe algún algoritmo que aproxime un cero de n ecuaciones polinomiales con n incógnitas, coeficientes complejos y que, además, tenga complejidad polinomial en media? (Beltrán y Pardo 2009)

Problema xvii de Smale, caso *real* (Smale 2000)

¿Existe algún algoritmo que aproxime un cero de n ecuaciones polinomiales con n incógnitas, coeficientes reales y que, además, tenga complejidad polinomial en media?

El problema xvii de Smale

Problema xvii de Smale (Smale 2000)

¿Existe algún algoritmo que aproxime un cero de n ecuaciones polinomiales con n incógnitas, coeficientes complejos y que, además, tenga complejidad polinomial en media? (Beltrán y Pardo 2009)

Problema xvii de Smale, caso *real* (Smale 2000)

¿Existe algún algoritmo que aproxime un cero de n ecuaciones polinomiales con n incógnitas, coeficientes reales y que, además, tenga complejidad polinomial en media?

El problema XVII de Smale, caso real

¿Por qué es importante?

- Problema clásico
- Resolución de problemas de optimización
- Modelos matemáticos no lineales como:
  - Redes eléctricas en corriente alterna
  - Equilibrios de Nash
  - Trayectorias de las partes de un robot

¿Por qué no valen los resultados anteriores?

- $\mathbb{R}^n$ es un espacio de medida nula en $\mathbb{C}^n$
- La geometría de las soluciones reales tiene muy malas propiedades (Azaïs y Wschebor 2005)
- Hay más de una componente conexa por caminos en el espacio de trabajo (Borges 2011)

El problema XVII de Smale, caso real

¿Por qué es importante?

- Problema clásico
- Resolución de problemas de optimización
- Modelos matemáticos no lineales como:
  - Redes eléctricas en corriente alterna
  - Equilibrios de Nash
  - Trayectorias de las partes de un robot

¿Por qué no valen los resultados anteriores?

- $\mathbb{R}^n$ es un espacio de medida nula en $\mathbb{C}^n$
- La geometría de las soluciones reales tiene muy malas propiedades (Azaïs y Wschebor 2005)
- Hay más de una componente conexa por caminos en el espacio de trabajo (Borges 2011)

Índice

Un algoritmo trivial

Sea $g : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ un sistema de ecuaciones polinomiales con coeficientes reales

Algoritmo Evolutivo Trivial

Estructura de Datos:

$$\mathbb{A} := \mathbb{R}^n$$

Operador de Calificación:

$$f\colon \mathbb{R}^n \to \mathbb{R}^n$$

$$x \mapsto \|g(x)\|^2$$

Operador de Evolución:

$$e\colon \mathbb{R}^{2n} \to \mathbb{R}^n$$

$$(x, y) \mapsto \frac{x + y}{2}$$

Condición de parada:

$$\|g(x)\|^2 < \varepsilon$$

Un algoritmo trivial

Sea $g : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ un sistema de ecuaciones polinomiales con coeficientes reales

Algoritmo Evolutivo Trivial

Estructura de Datos:

$$\mathbb{A} := \mathbb{R}^n$$

Operador de Calificación:

$$f\colon \mathbb{R}^n \to \mathbb{R}^n$$

$$x \mapsto \|g(x)\|^2$$

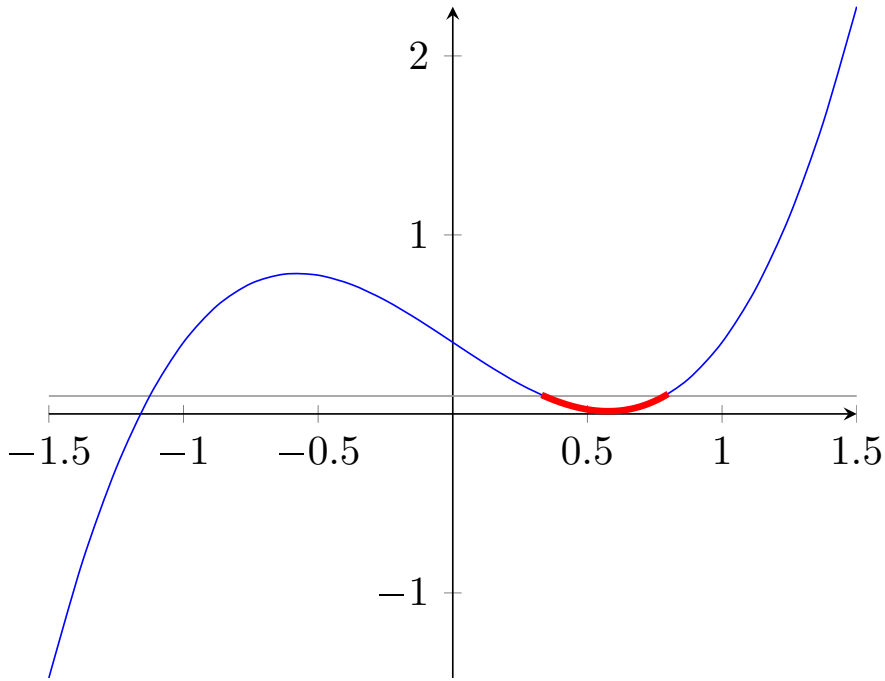
Operador de Evolución:

$$e\colon \mathbb{R}^{2n} \to \mathbb{R}^n$$

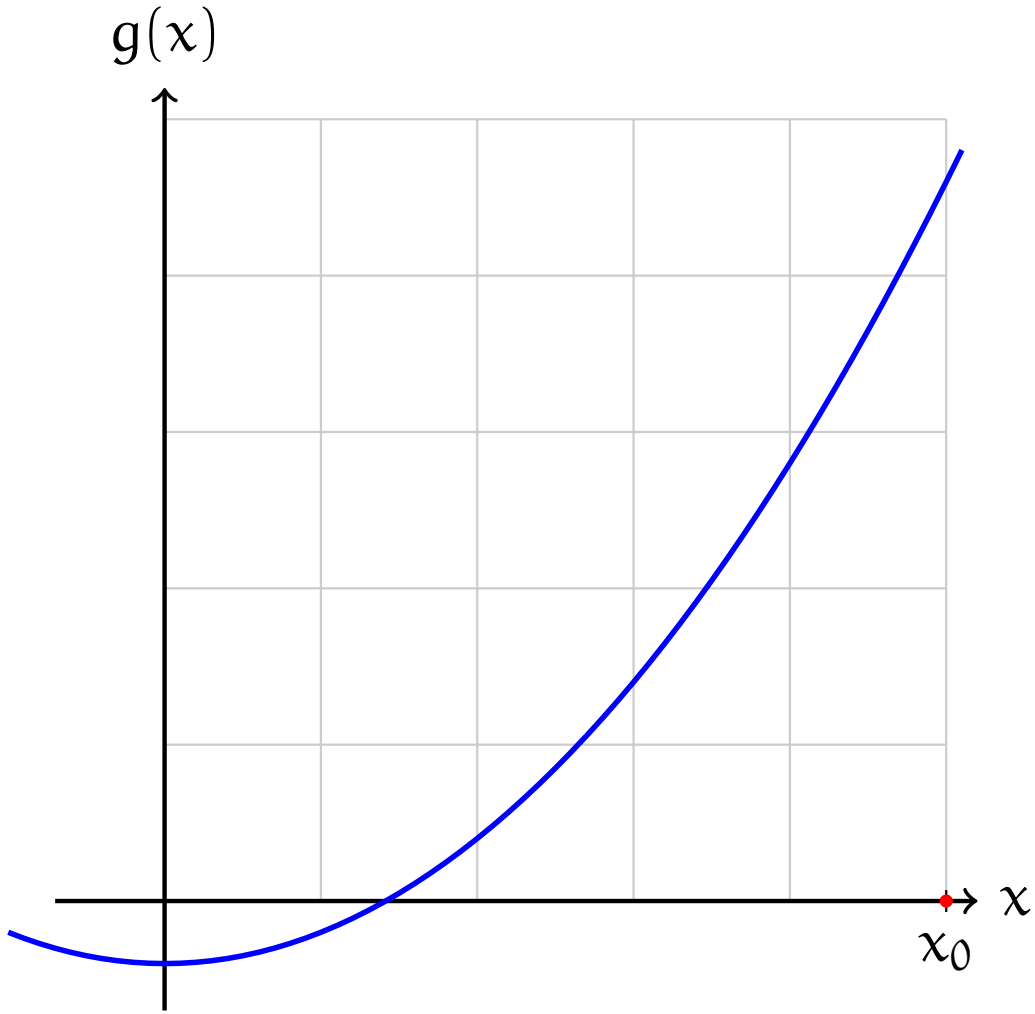
$$(x, y) \mapsto \frac{x + y}{2}$$

Condición de parada:

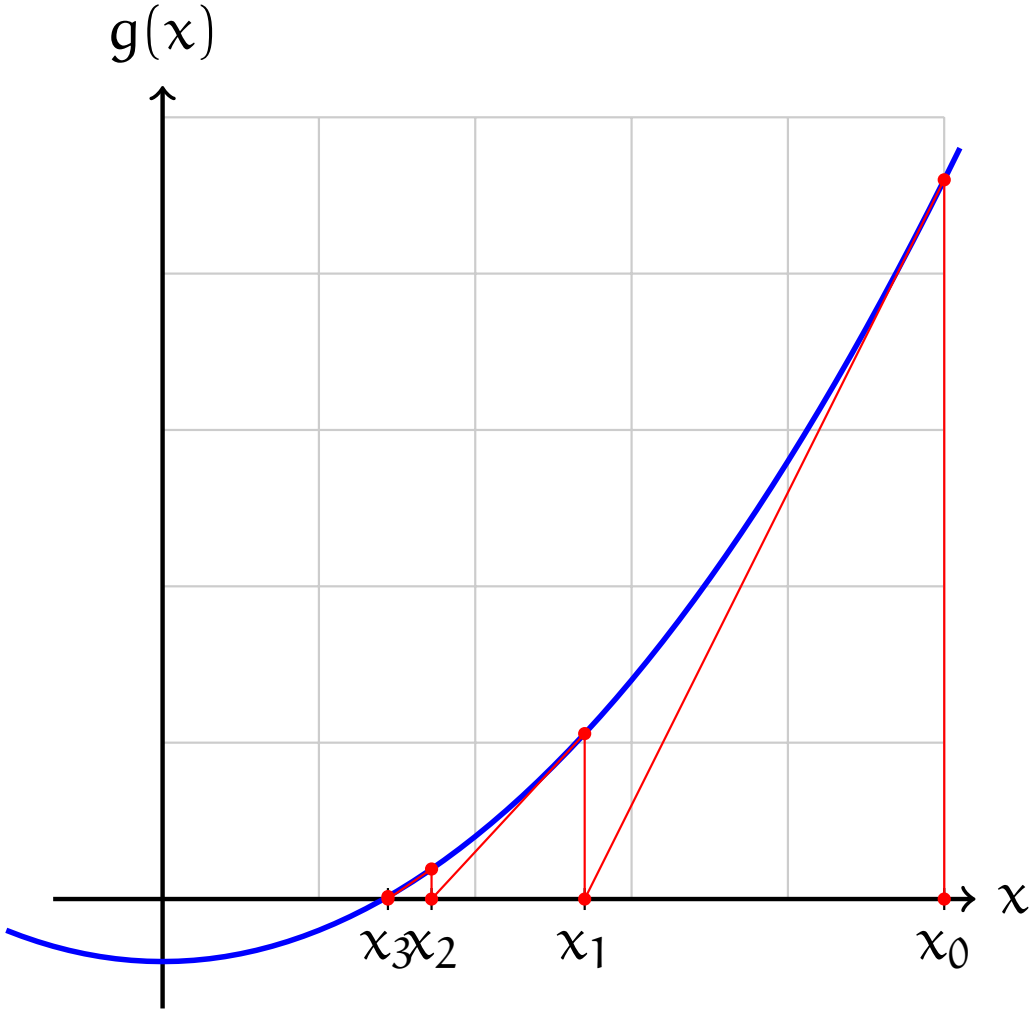
$$\|g(x)\|^2 < \varepsilon$$



Otra alternativa: el método de Newton



Otra alternativa: el método de Newton



Una extensión al programa de Shub-Smale

Los tres teoremas fundamentales

α^*-Teorema (Borges, Montaña y Pardo 2014)

Si $\alpha^*(g, x_0) < \alpha_0$ entonces el método de Newton converge

Distancia entre raíces $\mathfrak{h}$ (Borges y Pardo 2008)

La distancia mínima entre dos raíces cualesquiera del sistema es $\mathfrak{h} \in \mathcal{O}((\mathcal{D}N)^{-1/4})$

Caja de zapatos H (Borges y Pardo 2008)

Con probabilidad $1 - \varepsilon$ al menos una raíz del sistema tiene norma menor que $H \in \mathcal{O}(\mathcal{D}\varepsilon^{-2})$

Algoritmo exhaustivo

Caja de zapato H
Distancia entre raíces $\mathfrak{h}$

$\Longrightarrow$

Podemos construir un retículo $\mathcal{L}$ que tiene al menos un punto donde el método de Newton converge

Algoritmo exhaustivo

En particular, realizando la prueba α^* sobre todos los puntos del retículo $\mathcal{L}$ obtenemos un algoritmo probabilista

PROBLEMA

El cardinal de $\mathcal{L}$ es de orden $\mathcal{O}(\mathcal{D}^{\mathrm{n}}\mathrm{N}\varepsilon^{-2})$

SOLUCIÓN

¿Por qué tenemos que probar en todos?

Algoritmo exhaustivo

Caja de zapato H
Distancia entre raíces $\mathfrak{h}$

$\Longrightarrow$

Podemos construir un retículo $\mathcal{L}$ que tiene al menos un punto donde el método de Newton converge

Algoritmo exhaustivo

En particular, realizando la prueba α^* sobre todos los puntos del retículo $\mathcal{L}$ obtenemos un algoritmo probabilista

PROBLEMA

El cardinal de $\mathcal{L}$ es de orden $\mathcal{O}(\mathcal{D}^{\mathrm{n}}\mathrm{N}\varepsilon^{-2})$

SOLUCIÓN

¿Por qué tenemos que probar en todos?

Algoritmo exhaustivo

Caja de zapato H
Distancia entre raíces $\mathfrak{h}$

$\Longrightarrow$

Podemos construir un retículo $\mathcal{L}$ que tiene al menos un punto donde el método de Newton converge

Algoritmo exhaustivo

En particular, realizando la prueba α^* sobre todos los puntos del retículo $\mathcal{L}$ obtenemos un algoritmo probabilista

PROBLEMA

El cardinal de $\mathcal{L}$ es de orden $\mathcal{O}(\mathcal{D}^{\mathrm{n}}\mathrm{N}\varepsilon^{-2})$

SOLUCIÓN

¿Por qué tenemos que probar en todos?

Algoritmo exhaustivo

Caja de zapato H
Distancia entre raíces $\mathfrak{h}$

$\Longrightarrow$

Podemos construir un retículo $\mathcal{L}$ que tiene al menos un punto donde el método de Newton converge

Algoritmo exhaustivo

En particular, realizando la prueba α^* sobre todos los puntos del retículo $\mathcal{L}$ obtenemos un algoritmo probabilista

PROBLEMA

El cardinal de $\mathcal{L}$ es de orden $\mathcal{O}(\mathcal{D}^{\mathrm{n}}\mathrm{N}\varepsilon^{-2})$

SOLUCIÓN

¿Por qué tenemos que probar en todos?

Algoritmo evolutivo

Sea $g : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ un sistema de ecuaciones polinomiales con coeficientes reales

Algoritmo Evolutivo

Estructura de Datos:

$$\mathbb{A} := \mathbb{R}^n$$

Operador de Calificación:

$$f\colon \mathbb{R}^n \to \mathbb{R}^n$$
$$x \mapsto \alpha^*(f, x)$$

Operador de Evolución:

$$e\colon \mathbb{R}^{2n} \to \mathbb{R}^n$$
$$(x, y) \mapsto \frac{x + y}{2}$$

Condición de parada:

$$\alpha^*(f, x) < \alpha_0$$

Resultados experimentales

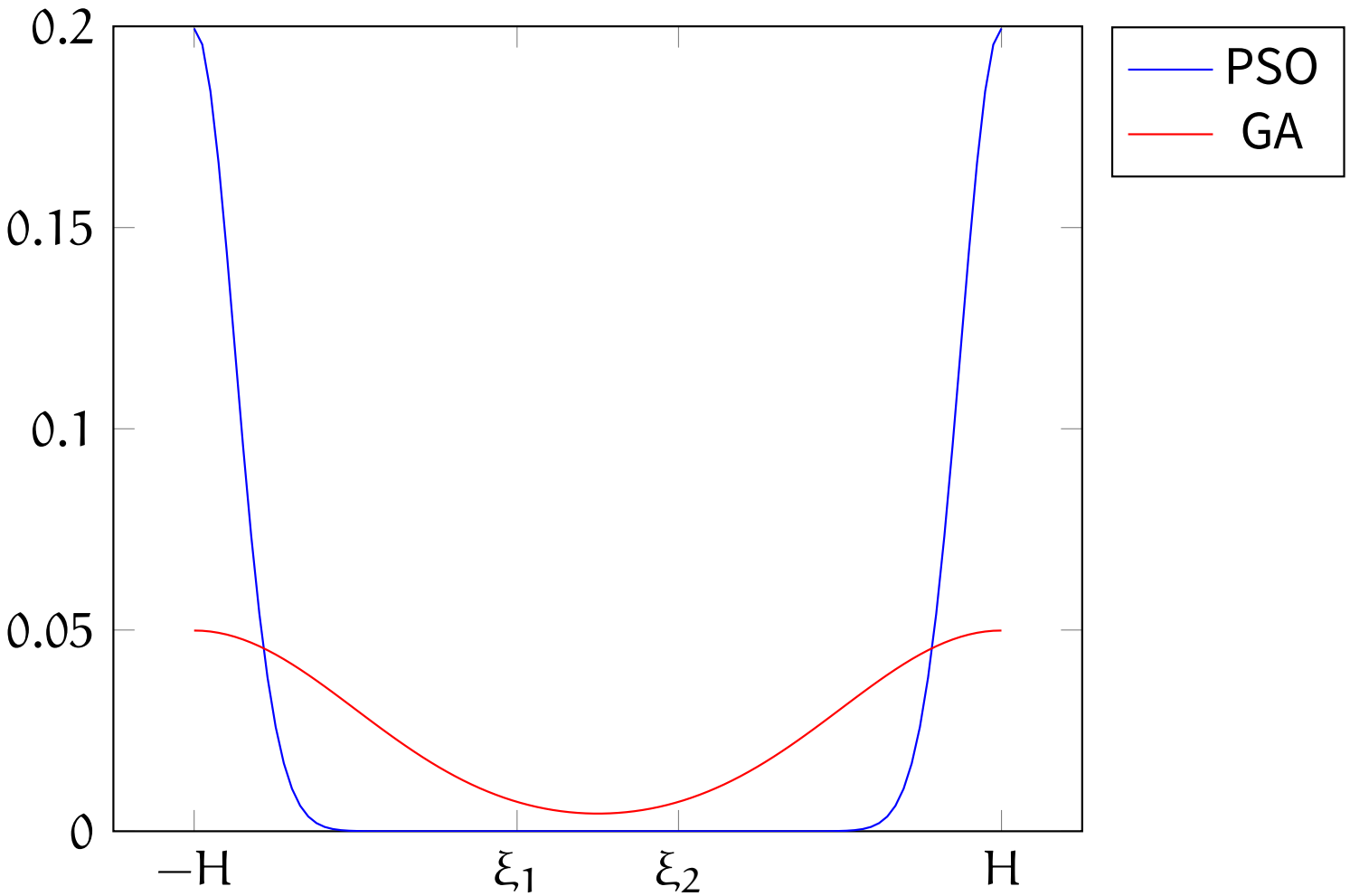
- Implementamos dos operadores de evolución tanto para la estrategia trivial como para la basada en α^*:
  - GA es el descrito anteriormente y
  - PSO basado en optimización por enjambre de partículas
- Repetimos cada experimento 100 veces
- Los resultados son el porcentaje de veces que se encuentra una raíz antes de 100 generaciones
- $dense_i$ son polinomios densos de i variables generados aleatoriamente
- $sparse_i$ son polinomios dados por slp de i variables generados aleatoriamente
- El resto son problemas de la base de datos Verschelde 2014
- $boon0$ es un sistema indeterminado
- $boonC$ no tiene raíces reales
- $fail$ tiene imagen con norma cercana a 0 [y sin raíces]

dataset	$\|\|f\|\| < 10^{-4}$		$\alpha_c^* < \alpha_0$	
	PSO	GA	PSO	GA
$dense_3$	100	74	100	75
$dense_5$	41	21	100	26
$dense_7$	33	1	100	2
$dense_{10}$	42	0	100	0
$sparse_3$	85	82	83	68
$sparse_5$	84	33	95	31
$sparse_7$	73	8	92	9
$sparse_{10}$	65	1	90	3
$chemequ$	100	100	100	100
$comb3000$	100	20	100	0
$i1$	73	13	100	15
ipp	100	56	100	60
$mickey$	100	91	100	91
$noon3$	100	96	100	96
$redcyc5$	86	80	100	80
$rediff3$	100	85	100	85
$sendra$	100	100	100	100
$boon0$	100	100	0	0
$boonC$	0	0	0	0
$fail$	0	40	0	0

Índice

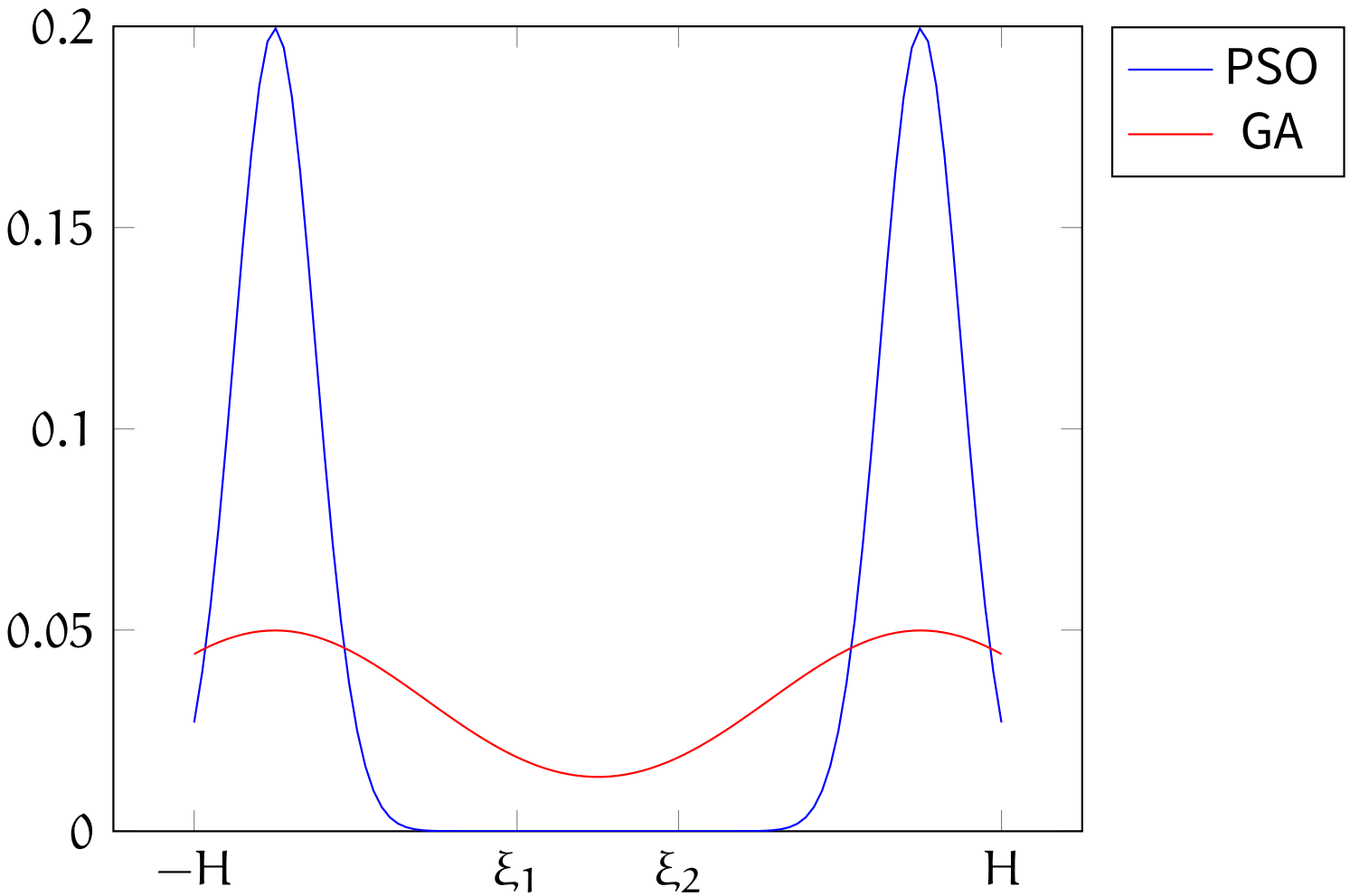
¿Qué [creo] está pasando?

Función de densidad de la distribución de la población



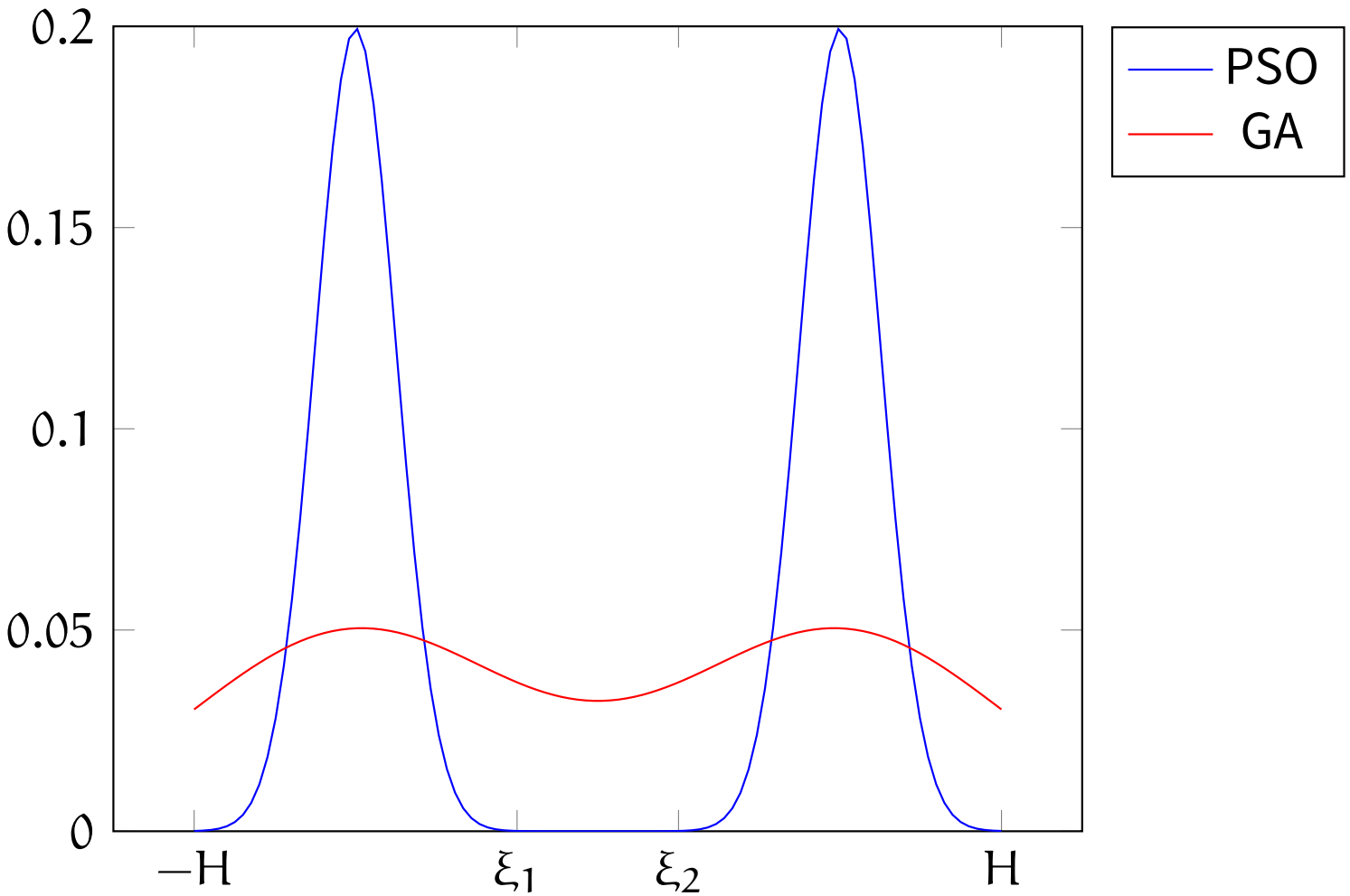
¿Qué [creo] está pasando?

Función de densidad de la distribución de la población



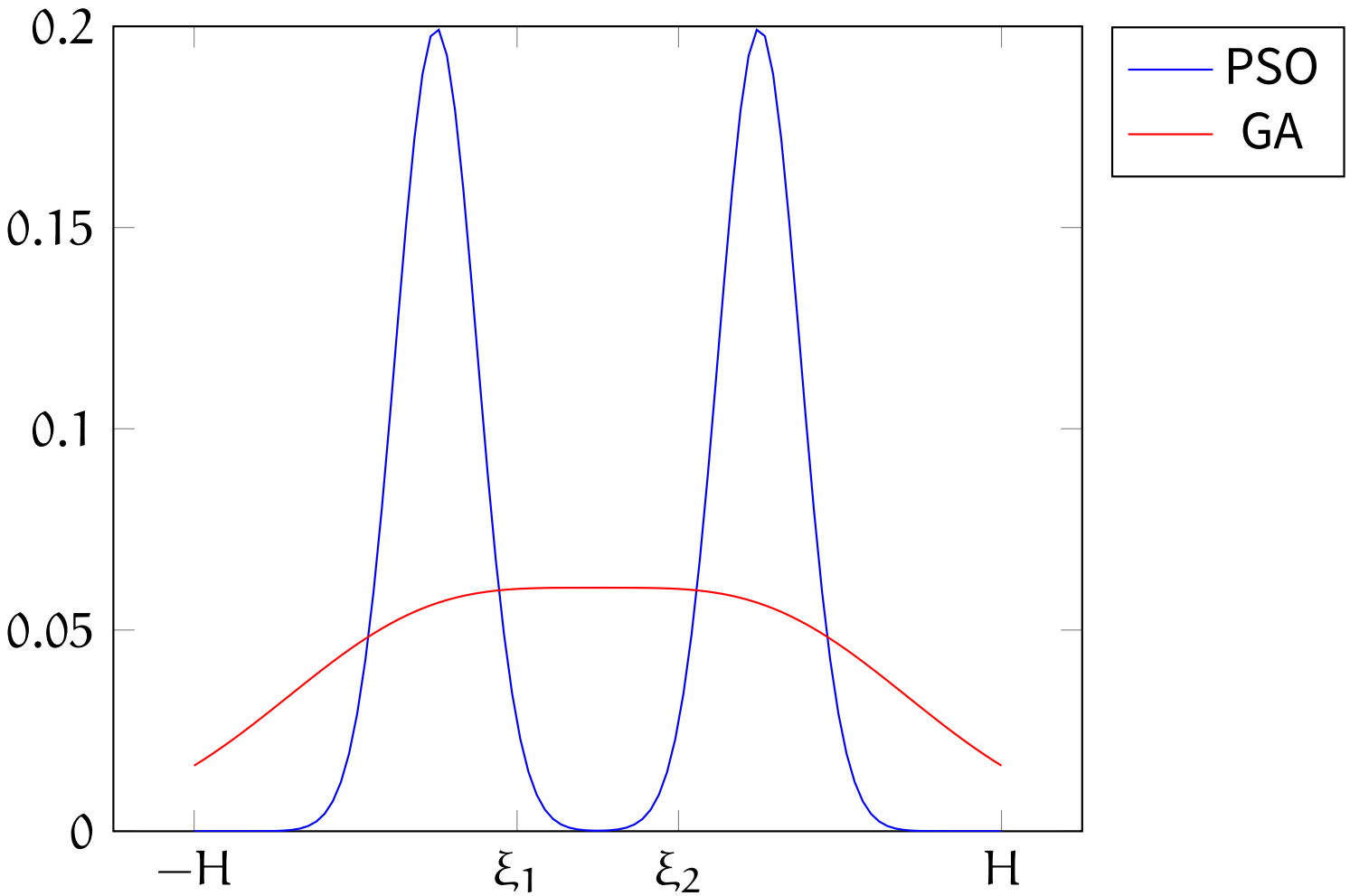
¿Qué [creo] está pasando?

Función de densidad de la distribución de la población



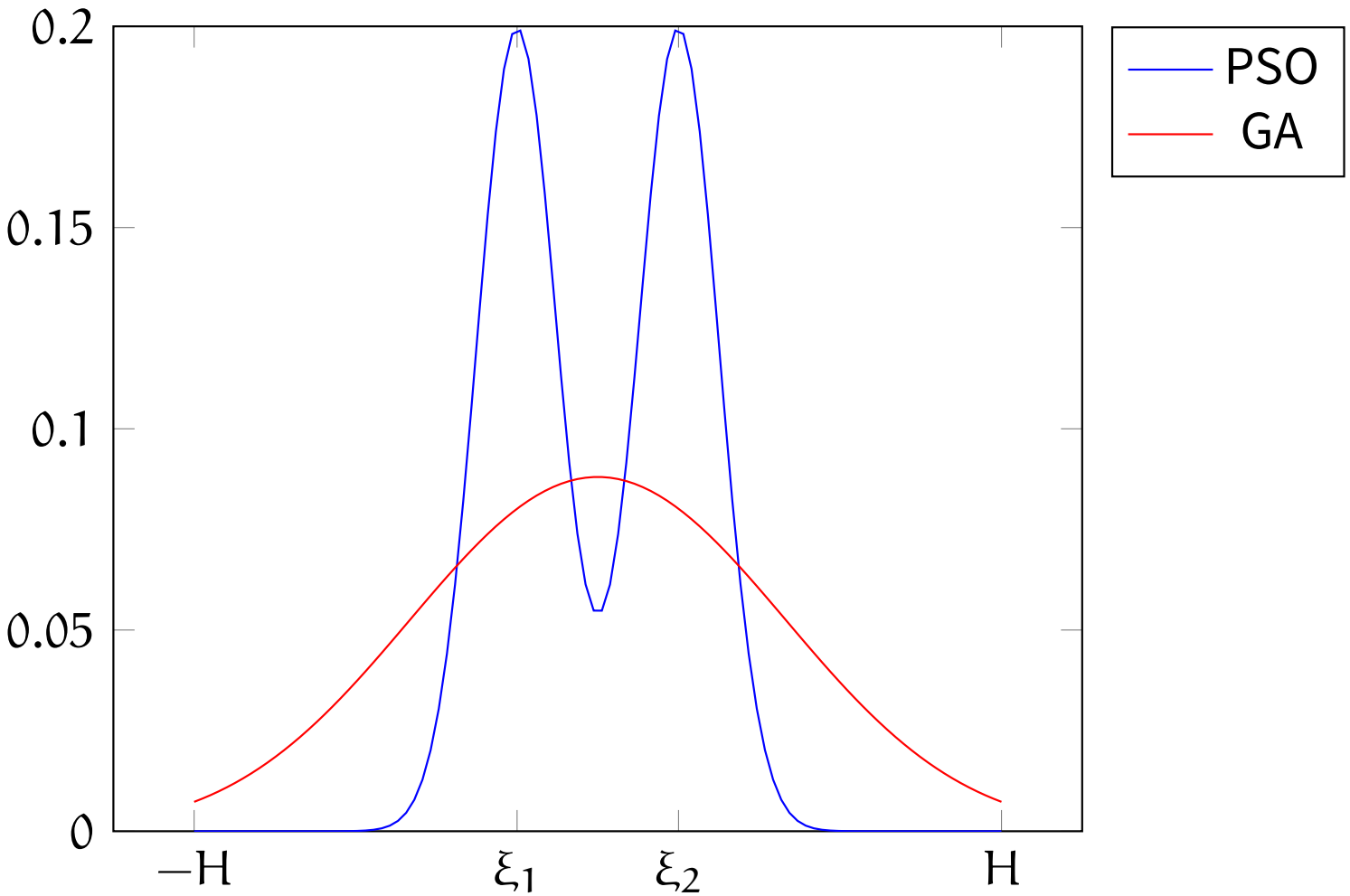
¿Qué [creo] está pasando?

Función de densidad de la distribución de la población



¿Qué [creo] está pasando?

Función de densidad de la distribución de la población

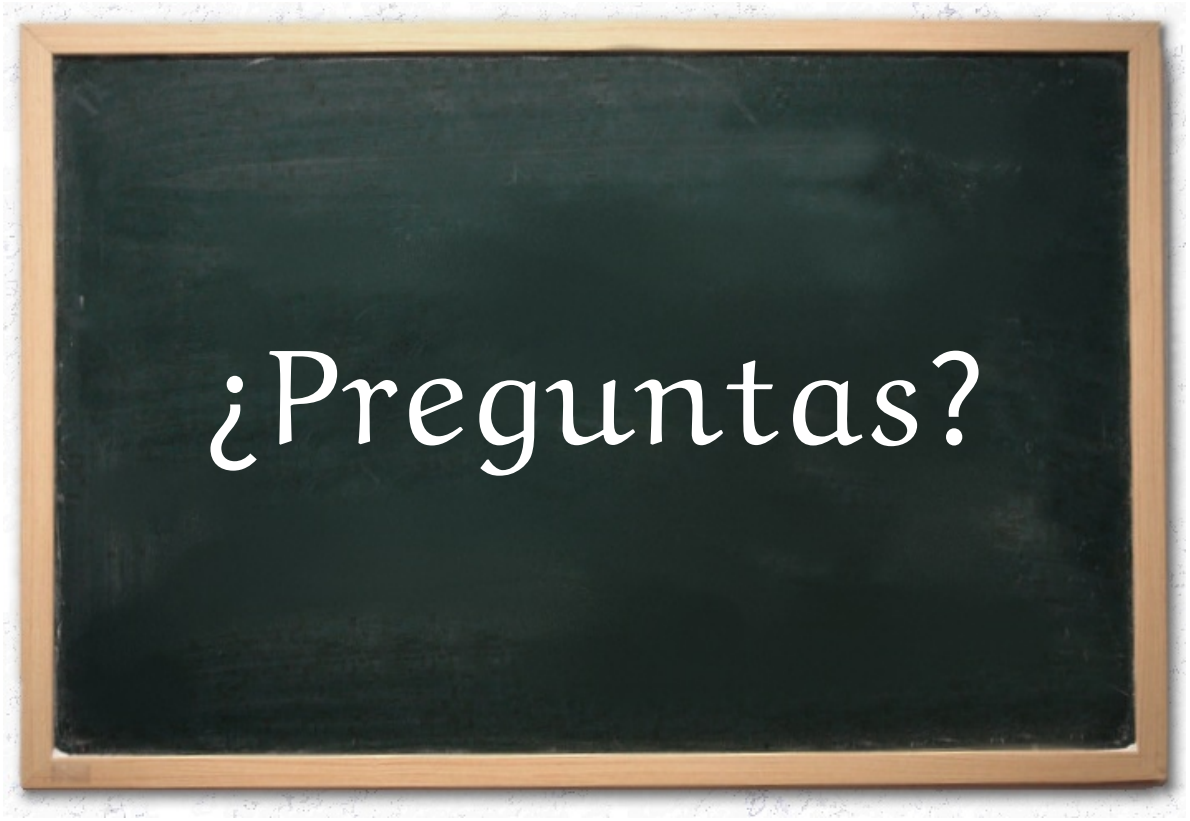


Trabajo futuro

Futuras acciones

1. Extender el modelo a un conjunto más general de funciones
2. Calcular y estudiar las propiedades de C_j
3. Optimizar el algoritmo evolutivo

Gracias por su atención



Bibliografía I

J. Azaïs y M. Wschebor.

«On the Roots of a Random System of Equations. The Theorem on Shub and Smale and Some Extensions».
En: *Found. Comput. Math.* 2 (2005), págs. 125-144.

C.E. Borges, J.L. Montaña y L.M. Pardo.

«A sharp fitness function for the problem of finding roots of polynomial equations systems».
En: *ECTA 2014 - Proceedings of the International Conference on Evolutionary Computation Theory and Applications*.
Rome, Italy, 2014,
Págs. 294-301.

C.E. Borges y L.M. Pardo.

«On the Probability Distribution of Data at Points in Real Complete Intersection Varieties».
En: *Journal of Complexity* 24.4 (2008), págs. 492-523.

C. Beltrán y L.M. Pardo.

«On Smale's 17 Problem: Average Polynomial Solver for Affine and Proyective Solutions».
En: *J.Amer. Math. Soc.* 22 (2009), págs. 363-385.

C.E. Borges.

«Programación Genética, Algoritmos Evolutivos y Aprendizaje Inductivo: Hacia una solución al problema XVII de Smale en el caso real».
Tesis doct. Universidad de Cantabria, 2011.

Bibliografía II

S. Smale.

«Mathematical Problems for the Next Century».
En: *Mathematics: Frontiers and Perspectives*.
Providence: Amer. Math. Soc., 2000,
Págs. 271-294.

Jan Verschelde.

The database of polynomial systems.
`http://homepages.math.uic.edu/~jan/demo.html`.
2014.

Certificando las soluciones dadas por algoritmos evolutivos

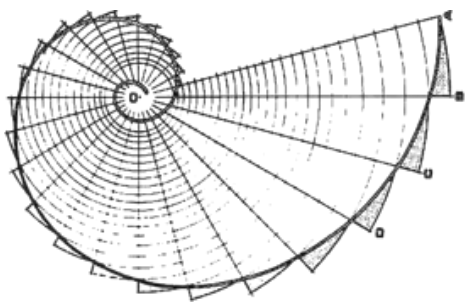
Cruz E. Borges
cruz.borges@deusto.es

Jose L. Montaña
montanjl@unican.es

Luis M. Pardo
luis.m.pardo@gmail.com

DeustoTech Energy

UC UNIVERSIDAD DE CANTABRIA



CONGRESO DE JÓVENES INVESTIGADORES

Real Sociedad Matemática Española

Universidad de Murcia, del 7 al 11 de Septiembre de 2015