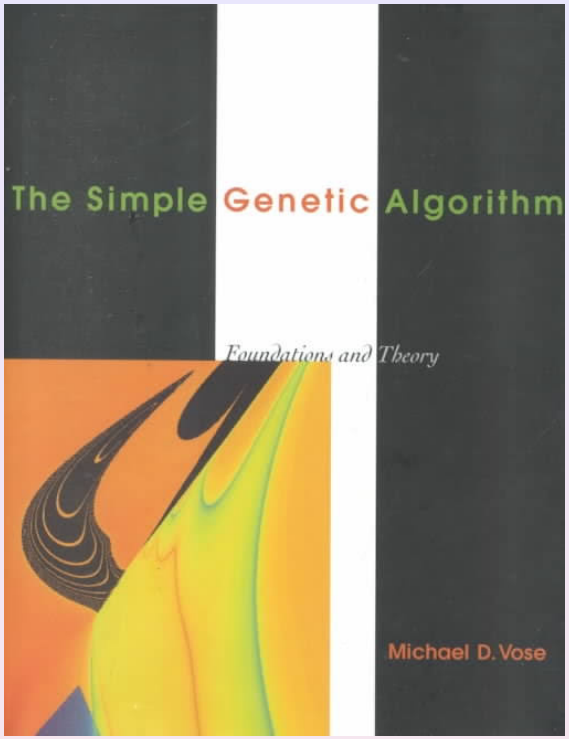
Introducción a la Computación Evolutiva

Carlos A. Coello Coello

carlos.coellocoello@ccinvestav.mx
CINVESTAV-IPN
Evolutionary Computation Group (EVOCINV)
Departamento de Computación
Av. IPN No. 2508, Col. San Pedro Zacatenco
México, D.F. 07360, MEXICO

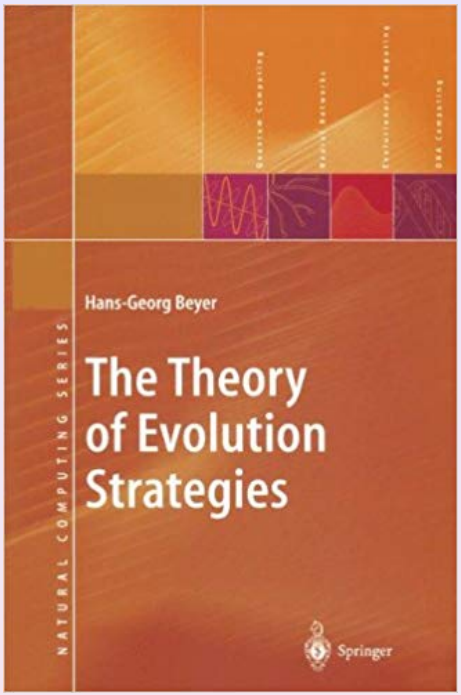
Clase 11

¿Cómo Funcionan los Algoritmos Genéticos?



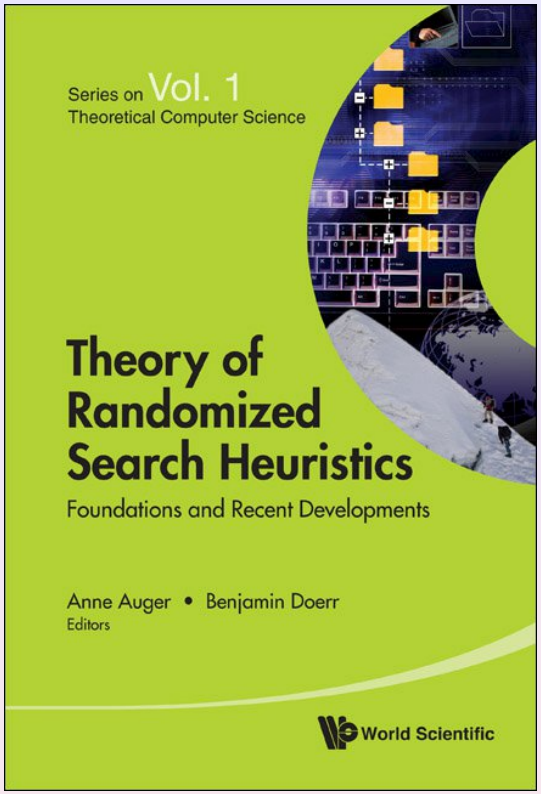
Aunque los algoritmos genéticos son simples de describir y programar, su comportamiento puede ser muy complicado, y todavía existen muchas preguntas abiertas acerca de cómo funcionan y sobre el tipo de problemas en los que son más adecuados.

¿Cómo Funcionan los Algoritmos Genéticos?



La teoría tradicional de los algoritmos genéticos (formulada originalmente por Holland en 1975) presupone que, en un nivel muy general de descripción, los algoritmos genéticos trabajan descubriendo, enfatizando y recombinando buenos "**bloques constructores**" de soluciones en una manera altamente paralelizada.

¿Cómo Funcionan los Algoritmos Genéticos?



La idea básica aquí es que las buenas soluciones tienden a estar formadas de buenos **bloques constructores** (combinaciones de bits que confieren una aptitud alta a las cadenas en las que están presentes).

¿Cómo Funcionan los Algoritmos Genéticos?

Holland [1975] introdujo la noción de esquemas para formalizar la noción de “bloques constructores”.

Un **esquema** es un conjunto de cadenas de bits que pueden describirse mediante una plantilla hecha de unos, ceros y asteriscos que funcionan como comodines (pueden tomar cualquier valor dentro del alfabeto en uso).

¿Cómo Funcionan los Algoritmos Genéticos?

Por ejemplo, el esquema **H = 1****1** representa el conjunto de todas las cadenas de 6 bits que empiezan y terminan con **1**.

Usaremos la notación **H** para denotar "**hiperplano**" [Goldberg, 1989], porque los esquemas definen *hiperplanos* ("planos" de varias dimensiones en el espacio *l*-dimensional de las cadenas de *l* bits).

David E. Goldberg, "**Genetic Algorithms in Search, Optimization and Machine Learning**", Addison-Wesley Publishing Co., Reading, Massachusetts, USA, 1989.

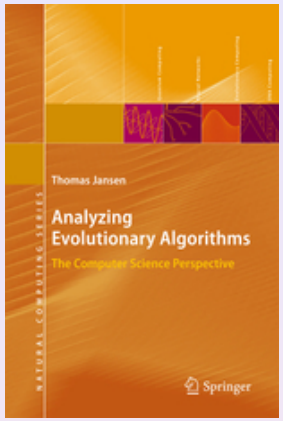
¿Cómo Funcionan los Algoritmos Genéticos?

Las cadenas que se ajustan a este esquema (por ejemplo, **100111** y **110011**) son instancias de **H**.

El esquema **H** tiene **2** bits definidos (posiciones que no son asteriscos), por lo que su **orden** es de **2**.

Su **longitud de definición**, denotada por δ (o sea, la distancia entre sus bits definidos más extremos) es **5**.

¿Cómo Funcionan los Algoritmos Genéticos?

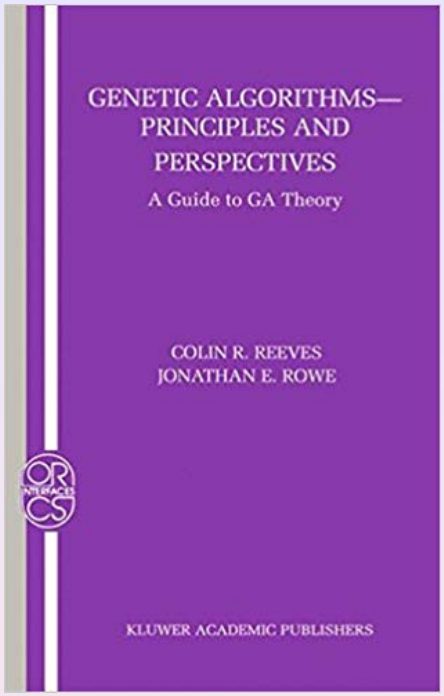


¿Cómo procesa un algoritmo genético los esquemas?

Cualquier cadena de bits dada de longitud l es una instancia de 2^l esquemas diferentes.

Por ejemplo, la cadena **11** es una instancia de ** (todas las cadenas binarias posibles de **2** bits), ***1**, **1*** y **11** (el esquema que contiene sólo la cadena **11**).

¿Cómo Funcionan los Algoritmos Genéticos?



En otras palabras, cualquier población dada de N cadenas contiene instancias de entre 2^l y $N \times 2^l$ esquemas diferentes.

Si todas las cadenas son idénticas, entonces hay 2^l instancias de exactamente 2^l esquemas diferentes; de lo contrario, el número es menor o igual a $N \times 2^l$.

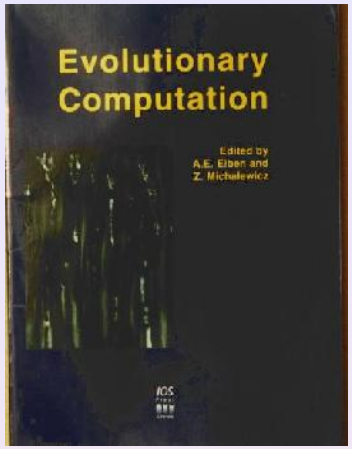
¿Cómo Funcionan los Algoritmos Genéticos?

Esto significa que, en una cierta generación, mientras el algoritmo genético está evaluando explícitamente las aptitudes de las *n* cadenas en la población, está también *estimando implícitamente* las aptitudes promedio de un número mucho mayor de esquemas.

En este caso, la aptitud promedio de un esquema se define como *la aptitud promedio de todas las instancias posibles de ese esquema en la población*.

A esta propiedad que presumiblemente tienen los algoritmos genéticos, se le conoce como **paralelismo implícito** [Holland, 1975].

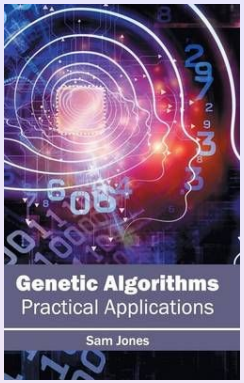
¿Cómo Funcionan los Algoritmos Genéticos?



Por ejemplo, en una población generada aleatoriamente de n cadenas, en promedio la mitad de las cadenas serán instancias de **1*****...*** y la otra mitad serán instancias de **0*****...***.

Las evaluaciones de las aproximadamente $n/2$ cadenas que son instancias de **1*****...*** nos da un estimado de la aptitud promedio de ese esquema (es un estimado porque las instancias evaluadas en poblaciones de tamaños típicos son sólo una pequeña muestra de todas las instancias posibles).

¿Cómo Funcionan los Algoritmos Genéticos?



Así como los esquemas no se representan o evalúan explícitamente por el algoritmo genético, los estimados de las aptitudes promedio de los esquemas **NO** se calculan o almacenan explícitamente por el algoritmo genético.

Sin embargo, el comportamiento del algoritmo genético, en términos del incremento y decremento en números de instancias de esquemas dados en la población, puede describirse como si realmente estuviera calculando y almacenando estos promedios.

¿Cómo Funcionan los Algoritmos Genéticos?

Podemos calcular la dinámica aproximada de este incremento y decremento en las instancias de los esquemas de la manera siguiente.

Hagamos que **H** sea un esquema con al menos una instancia presente en la población en la generación t.

Hagamos que $m(H,t)$ sea el número de instancias de **H** en la generación t, y que $\hat{u}(H,t)$ sea la **aptitud promedio observada** de **H** en la generación t (es decir, la aptitud promedio de las instancias de **H** en la población en la generación t).

Lo que queremos calcular es $E(m(H,t+1))$, o sea el número esperado de instancias de **H** en la generación $t+1$.

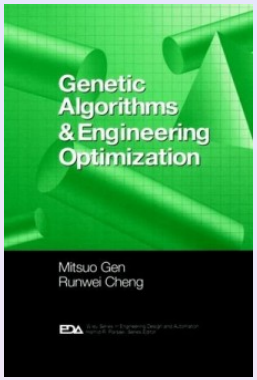
Supongamos que usamos **selección proporcional**.

Bajo este esquema, el número esperado de descendientes de una cadena x es igual a $f(x)/\bar{f}(t)$, donde $f(x)$ es la aptitud de x y $\bar{f}(t)$ es la aptitud promedio de la población en la generación t.

Entonces, suponiendo que x está en la población en la generación t, y haciendo que $x \in H$ denote "x es una instancia de **H**", e ignorando (por ahora) los efectos de la cruza y la mutación, tenemos:

$$E(m(H, t+1)) = \sum_{x \in H} f(x)/\bar{f}(t) = (\hat{u}(H, t)/\bar{f}(t))m(H, t) \qquad (1)$$

¿Cómo Funcionan los Algoritmos Genéticos?



Esta sustitución es posible por definición, puesto que

$$\hat{u}(H,t) = \left(\sum_{x \epsilon H} f(x) \right) / m(H,t)$$

para una x que se encuentre en la población en la generación t.

De tal forma que aunque el algoritmo genético no calcule explícitamente $\hat{u}(H,t)$, los incrementos o decrementos de las instancias de esquemas en la población dependen de esta cantidad.

¿Cómo Funcionan los Algoritmos Genéticos?

Tanto la cruza como la mutación destruyen y crean instancias de **H**.

Por ahora incluiremos sólo los efectos destructivos de la cruza y la mutación (es decir, aquellos que decrementan el número de instancias de **H**).

Si incluimos estos efectos, modificamos el lado derecho de la ecuación (1) para dar un límite inferior de $E(m(H, t+1))$.

Hagamos que p_c sea la probabilidad de cruza (de un punto) y supongamos que una instancia del esquema **H** se selecciona para ser padre.

El esquema **H** se dice que **sobrevive** bajo la cruza de un punto si uno de sus hijos es también una instancia del esquema **H**.

¿Cómo Funcionan los Algoritmos Genéticos?

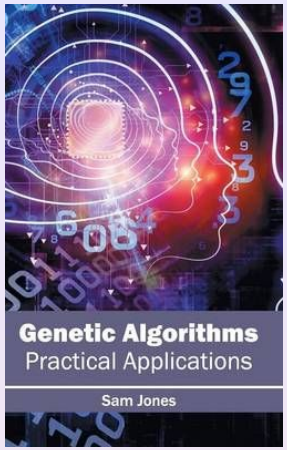
Podemos proporcionar un límite inferior de la probabilidad $S_c(H)$ de que **H** sobrevivirá la cruza de un punto:

$$S_c(H) \geq 1 - p_c \left(\frac{\delta(H)}{l-1} \right)$$

donde $\delta(H)$ es la longitud de definición de **H** y l es la longitud de las cadenas en el espacio de búsqueda.

Esto es, las cruzas que ocurren dentro de la longitud de definición de **H** pueden destruir a **H** (es decir, pueden producir hijos que no son instancias de **H**), así que multiplicamos la fracción de la cadena que **H** ocupa por la probabilidad de cruza para obtener un límite superior de la probabilidad de que el esquema será destruído.

¿Cómo Funcionan los Algoritmos Genéticos?



El valor es un límite superior porque algunas cruzas dentro de las posiciones definidas de un esquema no lo destruirán (por ejemplo, si dos cadenas idénticas se cruzan).

Al sustraer este valor de 1 obtenemos un límite inferior.

En resumen, la probabilidad de supervivencia bajo el efecto del operador de cruza es alta para esquemas más cortos.

¿Cómo Funcionan los Algoritmos Genéticos?

Ahora cuantificaremos los efectos de perturbación de la mutación.

Hagamos que p_m sea la probabilidad de mutación y que $S_m(H)$ sea la probabilidad de que el esquema **H** sobrevivirá bajo la m utación de una instancia de **H**.

Esta probabilidad es igual a:

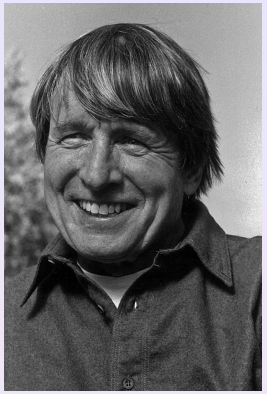
$$(1 - p_m)^{o(H)}$$

donde $o(H)$ es el orden de **H** (es decir, el número de bits definidos en **H**).

Esto es, para cada bit, la probabilidad de que el bit no se mutará es $1 - p_m$, de manera que la probabilidad de que bits no definidos del esquema **H** se muten es esta cantidad multiplicada por sí misma $o(H)$ veces.

En resumen, la probabilidad de supervivencia bajo mutación es más alta para esquemas de menor orden.

¿Cómo Funcionan los Algoritmos Genéticos?



Estos efectos de perturbación pueden usarse para modificar la ecuación (1):

$$E(m(H,t+1)) \geq \frac{\hat{u}(H,t)}{\vec{f}(t)} m(H,t) \left(1 - p_c \frac{\delta(H)}{l-1} \right) [(1-p_m)^{o(H)}]$$

A esta expresión se le conoce como el **Teorema de los Esquemas** [Holland, 1975], y describe el crecimiento de un esquema de una generación a la siguiente.

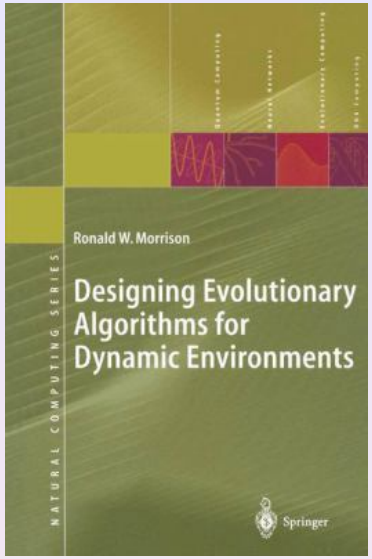
¿Cómo Funcionan los Algoritmos Genéticos?

La interpretación del **Teorema de los Esquemas** es la siguiente: "los esquemas cortos y de bajo orden, cuya aptitud promedio se mantiene por encima de la media de aptitud (de la población), recibirán un número mayor de muestras conforme transcurran las generaciones."

El **Teorema de los Esquemas** es un límite inferior puesto que lidia solamente con los efectos destructivos de la cruza y la mutación.

Sin embargo, se cree que la cruza es la fuente de mayor poder del algoritmo genético, pues tiene la capacidad de recombinar las instancias de esquemas favorables para formar otros igualmente buenos o mejores.

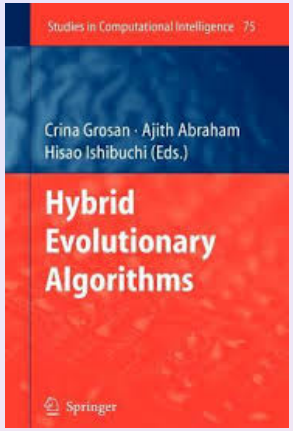
¿Cómo Funcionan los Algoritmos Genéticos?



A partir del Teorema de los Esquemas, se puede formular la denominada **Hipótesis de los Bloques Constructores** [Goldberg, 1989]:

Un algoritmo genético realiza la búsqueda a través de las yuxtaposición de esquemas cortos, de bajo orden y elevada aptitud, a los que se denomina **bloques constructores**.

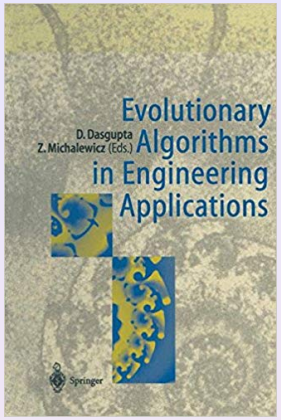
¿Cómo Funcionan los Algoritmos Genéticos?



El efecto de la selección es sesgar gradualmente el procedimiento de muestreo hacia instancias de esquemas cuya aptitud se estime estén sobre el promedio.

Con el paso del tiempo, el estimado de la aptitud promedio de un esquema debiera, en principio, volverse cada vez más preciso puesto que el algoritmo genético está muestreando más y más instancias de este esquema.

¿Cómo Funcionan los Algoritmos Genéticos?



El Teorema de los Esquemas y la Hipótesis de los Bloques Constructores lidian primordialmente con el papel de la selección y la cruza en los algoritmos genéticos, pero ¿cuál es el papel de la mutación?

Holland [1975] propuso que la mutación previene la pérdida de diversidad en una posición cualquiera. Es una especie de "póliza de seguro" contra fijaciones en una cadena cromosómica.

Críticas al Teorema de los Esquemas

El Teorema de los Esquemas es realmente una desigualdad "débil", no un "teorema".

Las siguientes afirmaciones sobre el teorema de los esquemas no son del todo demostrables:

(a) Los esquemas por arriba del promedio se incrementan exponencialmente con el tiempo (afirmación hecha en el libro de Goldberg [1989]).

(b) Los esquemas por arriba del promedio se exploran rápidamente en paralelo sin alentar de manera significativa la búsqueda (interpretación errónea del paralelismo implícito).

(c) Aproximadamente, se procesan n^3 esquemas de manera útil y en paralelo por cada generación (afirmación hecha en el libro de Goldberg [1989]).

No Free Lunch Theorem

Fue formulado por David Wolpert y William MacReady (del Instituto Santa Fe) en 1996, y se publicó en 1997.

En términos breves y simples, en este trabajo se demuestra que todas las técnicas de búsqueda heurística son matemáticamente equivalentes en general.

Es decir, se demuestra (matemáticamente) que no es posible diseñar una técnica heurística que pueda superar a todas las demás en todas las clases de problemas.

David H. Wolpert and William G. Macready, "**No Free Lunch Theorems for Optimization**", *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, pp. 67–82, April 1997.

No Free Lunch Theorem

Moraleja: el énfasis que suele ponerse en optimizar con técnicas heurísticas (como el algoritmo genético) es erróneo.

¿Qué alternativa tenemos entonces? Investigar el comportamiento emergente de una técnica heurística.

¿Cuál es el costo de esta alternativa? Formalizar nuestro modelo heurístico y realizar demostraciones a partir de dicha formalización.

¿Qué ganamos? Una comprensión conceptual de la técnica y una descripción a fondo de las circunstancias en las cuales un algoritmo genético es la mejor alternativa.

Ejemplo de un Problema Deceptivo

Supongamos que tenemos una función de aptitud que nos devuelve los siguientes valores para las cadenas binarias de longitud 3:

Cadena	Aptitud
000	70
001	50
010	49
011	1
100	30
101	2
110	3
111	80

Las cadenas con mayor número de ceros tienen mayor aptitud, pero el óptimo global es la cadena de todos unos.

En este caso, el algoritmo genético tenderá a favorecer durante la selección a las cadenas con más ceros y encontrará la cadena de todos ceros (un óptimo local) en vez de llegar al óptimo global.

Fundamentos Teóricos de los Algoritmos Genéticos

Algunas de las preguntas más importantes que se han planteado dentro de la comunidad de los algoritmos genéticos son las siguientes:

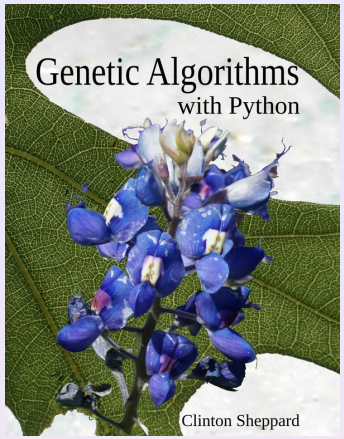
1. ¿Qué leyes describen el comportamiento macroscópico de los algoritmos genéticos? En particular, ¿qué predicciones pueden hacerse acerca del cambio de aptitud en el tiempo y acerca de las dinámica de la población en un algoritmo genético en particular?
2. ¿Cómo dan lugar los operadores de bajo nivel (selección, cruza y mutación) al comportamiento macroscópico de los algoritmos genéticos?
3. ¿En qué tipo de problemas es más probable que los algoritmos genéticos tengan un buen desempeño?

4. ¿En qué tipo de problemas es más probable que los algoritmos genéticos tengan un mal desempeño?
5. ¿Qué significa para un algoritmo genético tener un “buen desempeño” o un “mal desempeño” ? Esto es, ¿qué criterios de desempeño son apropiados para un algoritmo genético?
6. ¿Bajo qué condiciones (tipos de algoritmos genéticos y tipos de problemas) superará un algoritmo genético a otras técnicas de búsqueda tales como escalando la colina y los métodos de gradiente?

El algoritmo genético suele considerarse una técnica que es buena para encontrar rápidamente regiones prometedoras del espacio de búsqueda, pero para realizar verdaderamente optimización se ha demostrado que en muchas instancias los híbridos de un algoritmo genético con otra técnica (por ejemplo, escalando la colina o un método de gradiente) suelen obtener mejores resultados.

Aunque los algoritmos genéticos pueden encontrar los óptimos globales de problemas de alta complejidad, la realidad es que muchas veces el costo computacional que requieren es prohibitivamente alto, y se prefieren para encontrar una solución razonable, ya que eso suelen poder hacerlo en un tiempo relativamente corto.

Fundamentos Teóricos de los Algoritmos Genéticos



Como heurística, el algoritmo genético no resulta muy adecuado para problemas críticos en los cuales el no encontrar una solución en un período de tiempo muy corto puede causar fallas irreversibles al sistema.

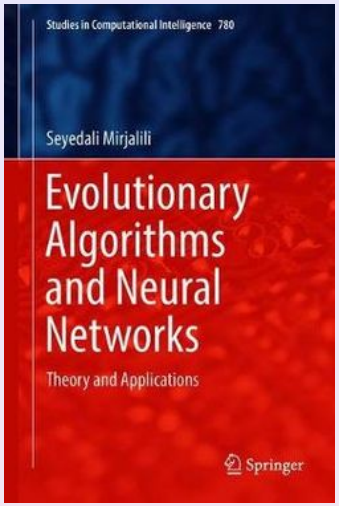
Asimismo, el algoritmo genético no es apropiado para aplicaciones en tiempo real en las que la respuesta debe proporcionarse de manera inmediata conforme se interactúa con el ambiente.

Fundamentos Teóricos de los Algoritmos Genéticos

Algunos autores (por ejemplo, Kenneth De Jong) han argumentado elocuentemente que, contrario a lo que se cree, los algoritmos genéticos no son optimizadores, sino que más bien son satisfactores de metas (o decisiones) secuenciales que pueden modificarse para actuar como optimizadores de funciones.

Kenneth A. De Jong, "**Genetic Algorithms are NOT Function Optimizers**", in L. Darrell Whitley (Editor), *Foundations of Genetic Algorithms 2*, pp. 5–17, Morgan Kaufmann Publishers, San Mateo, California, USA, 1993.

Fundamentos Teóricos de los Algoritmos Genéticos



Si bien en la práctica han tenido un gran éxito como optimizadores, la realidad es que los algoritmos suelen tener dificultades para encontrar máximos globales en ciertas clases de problemas (como por ejemplo, el problema del viajero) sumamente susceptibles a la representación o aquellos en los que la evaluación de la función de aptitud resulta sumamente costoso (computacionalmente hablando).

Fundamentos Teóricos de los Algoritmos Genéticos

El teorema de los esquemas se ha usado por algunos en la comunidad de los algoritmos genético para proponer una respuesta a la pregunta: "**¿qué hace que un problema sea difícil para un algoritmo genético?**"

La comunidad de computación evolutiva suele suponer que la competencia entre los esquemas procede aproximadamente de particiones de esquemas de bajo orden a particiones de esquemas de orden alto.

A partir de esta premisa, Bethke [1981] infirió que sería difícil para un algoritmo genético encontrar el óptimo de una función de aptitud si las particiones de bajo orden contenían información errónea acerca de las particiones de orden más alto.

A.D. Bethke, "**Genetic Algorithms as Function Optimizers**", PhD thesis, Department of Computer and Communication Sciences, University of Michigan, Ann Arbor, Michigan, USA, 1981.

¿Qué es lo que hace difícil un problema para un algoritmo genético?

Consideremos un ejemplo, un tanto extremo, de lo dicho en el punto anterior.

Llamemos al esquema **H** "ganador" si su aptitud estática promedio es la más alta en su partición.

Supongamos que cualquier esquema cuyos bits definidos sean todos unos es un ganador, excepto por el esquema de longitud L correspondiente a $1111\ldots1$, y hagamos que $0000\ldots0$ sea un ganador.

En principio, será difícil para un algoritmo genético encontrar 0000...0, porque toda partición de bajo orden da información equivocada sobre dónde es más probable encontrar el óptimo.

¿Qué es lo que hace difícil un problema para un algoritmo genético?

A este tipo de función de aptitud se le llama "**con decepción total**".

Afortunadamente, nadie ha podido encontrar un problema del mundo real que exhiba **decepción total**. Sólo se han encontrado problemas reales con **decepción parcial**.

También es posible definir funciones de aptitud con menores cantidades de decepción (es decir, algunas particiones dan información incorrecta acerca de la localización del óptimo).

Bethke [1981] usó la "**Transformada de Walsh**" (análoga a la transformada de Fourier) para diseñar funciones de aptitud con varios grados de decepción.

¿Qué es lo que hace difícil un problema para un algoritmo genético?

La transformada de Walsh constituye un medio fácilmente manipulable de análisis de paisajes de búsqueda binarios, con el valor agregado de que hay una correspondencia natural entre las particiones de Walsh y los esquemas.

Hoy en día, la investigación en torno a la decepción ha disminuido considerablemente, aunque existen todavía algunos investigadores interesados en esta área.

Se hace notar, sin embargo, que varios investigadores han cuestionado la relevancia del análisis de los esquemas en la comprensión del funcionamiento verdadero de los algoritmos genéticos [Grefenstette, 1993; Mason, 1993; Peck, 1995].

¿Qué es lo que hace difícil un problema para un algoritmo genético?

Por ejemplo, Grefenstette [1993] afirma que mucho del trabajo efectuado en teoría de los algoritmos genéticos ha asumido un versión más fuerte de lo que él llama la "**Hipótesis Estática de los Bloques Constructores**" (HEBC):

> *Dada cualquier partición de esquemas de bajo orden y reducida longitud de definición, se espera que un algoritmo genético converja al hiperplano (en esa partición) con la mejor aptitud promedio estática (el 'ganador esperado').*

John J. Grefenstette, "**Deception Considered Harmful**", in L. Darrell Whitley (Editor), *Foundations of Genetic Algorithms 2*, pp. 75–91, Morgan Kaufmann Publishers, San Mateo, California, USA, 1993.

¿Qué es lo que hace difícil un problema para un algoritmo genético?

Esta formulación es más fuerte que la original, puesto que dice que el algoritmo genético convergerá a los "**verdaderos**" ganadores de cada competencia entre particiones cortas y de bajo orden en vez de que converja a esquemas con la mejor aptitud **observada**.

La HEBC no es lo que propuso Holland [1975], y nunca se ha demostrado o validado empíricamente, pero implícitamente involucra la premisa de que las funciones de aptitud con decepción serán difíciles para un algoritmo genético.

John H. Holland, **Adaptation in Natural and Artificial Systems**, University of Michigan Press, Ann Arbor, Michigan, USA, 1975.

¿Qué es lo que hace difícil un problema para un algoritmo genético?

Grefenstette proporciona 2 razones posibles por las que la HEBC puede fallar:

1. **Convergencia Colateral**: Una vez que la población comienza a converger hacia algún punto, las muestras de algunos esquemas dejarán de ser uniformes.

Por ejemplo, supongamos que las instancias de $111 * \ldots *$ son muy aptas y que la población ha convergido más o menos a esos bits (o sea, casi todos los miembros de la población son una instancia de ese esquema).

En este caso, casi todas las muestras de, por ejemplo, $***000 * \ldots *$, serán realmente muestras de $111000 * \ldots *$.

Esto puede impedir que el algoritmo genético haga una estimación precisa de $\hat{u}(***000 * \ldots *)$. Aquí la $\hat{u}(H)$ denota la **aptitud promedio estática** de un esquema **H** (el promedio sobre todas las instancias del esquema en el espacio de búsqueda).

¿Qué es lo que hace difícil un problema para un algoritmo genético?

2. **Elevada Varianza de la Aptitud**: Si la aptitud promedio estática de un esquema tiene una varianza alta, el algoritmo genético podría no ser capaz de efectuar una estimación precisa de esta aptitud promedio estática.

Consideremos, por ejemplo, la siguiente función de aptitud:

$$f(x) = \begin{cases} 2 & \text{si } x \in 111 * ... * \\ 1 & \text{si } x \in 0 * ... * \\ 0 & \text{de lo contrario} \end{cases}$$

La varianza de $1 * ... *$ es alta, así que el algoritmo genético converge a las subregiones de aptitud elevada de este esquema.

Esto sesga a todas las muestras subsecuentes de este esquema, impidiendo que se puede obtener un estimado preciso de su aptitud estática.

¿Qué es lo que hace difícil un problema para un algoritmo genético?

Esto tiene relación directa con la importancia de la decepción en el comportamiento de los algoritmos genéticos, porque las funciones de aptitud con decepción se definen completamente en términos de las aptitudes estáticas promedio de los esquemas.

Para ilustrar su argumento, Grefenstette [1993] da ejemplos de problemas con decepción que son fáciles de optimizar con un algoritmo genético y de problemas que no tienen decepción y que son arbitrariamente difíciles para un algoritmo genético.

Su conclusión es de que la decepción no es una causa necesaria ni suficiente para que un algoritmo genético tenga dificultades, y de que su relevancia en el estudio de los algoritmos genéticos debe ser demostrada todavía.