

Un Ambiente Gráfico para Testear Estrategias Evolutivas en el Problema Inverso de las IFS

María Laura Ivanissevich¹, Paula Millado¹, Luis Sierpe¹ y Claudio Delrieux²

¹Univ. Nacional de la Patagonia Austral, L. de la Torre 1070, (9300) Río Gallegos,

²Universidad Nacional del Sur, Alem 1253, (8000) Bahía Blanca, ARGENTINA.

Voice: (54)(291)4595101 ext. 3381 — Fax: (54)(291)4595154 — e-mail: claudio@acm.org

Palabras Clave: SISTEMAS DE FUNCIONES ITERADAS (IFS), PROGRAMACIÓN EVOLUTIVA, PROGRAMACIÓN GENÉTICA.

1 Introducción

Los sistemas de funciones iteradas (IFS) constituyen una manera de representar imágenes por medio de conjuntos fractales. La factibilidad de representar imágenes con IFS se fundamenta en el Teorema del Collage, según el cual una imagen cualquiera puede ser arbitrariamente aproximada por un atractor fractal adecuadamente representado por un IFS. Esta propiedad hace tentadora la idea de buscar un método de compresión de imágenes que encuentre el collage adecuado para cualquier imagen de entrada. Esto constituye el problema inverso del IFS. Sin embargo, la búsqueda exhaustiva de un IFS para aproximar una imagen dada está fuera de las posibilidades de los sistemas de cómputo actuales. Un IFS como el del helecho consta de 24 parámetros, por lo que encontrar sus respectivos valores por “fuerza bruta” requeriría un tiempo de cómputo imposible de cuantificar.

Las estrategias hasta ahora propuestas para solucionar esta dificultad son dos. Por un lado, es posible restringir el espacio de búsqueda a un subconjunto de las transformaciones afines, como se propone en la compresión fractal en bloques (CFB) [4, 6]. En la CFB se utiliza un conjunto relativamente grande de mapas entre segmentos de la imagen, utilizando escalas fijas, y rotaciones cuantizadas a cuartos de circunferencia. Por lo tanto, el espacio de búsqueda es relativamente pequeño para cada transformación. Los resultados obtenidos con la CFB poseen una buena relación de compromiso entre tiempo de cómputo, compresión resultante, y calidad final, aunque están evidentemente lejos de las posibilidades teóricas [5].

Por otro lado, es posible utilizar un mecanismo de búsqueda adaptativo, que tenga cierta inteligencia como para guiarse dentro del un subconjunto del espacio de búsqueda en el cual los resultados se acercan a la solución buscada. Este es el tipo de estrategia buscada con la programación evolutiva (PE) [8] y con la programación genética (PG) [7]. El uso de PE y PG para la solución del problema inverso ha sido mucho menos estudiado, pese a que permitiría obtener mejores compresiones, tal vez con mayores tiempos de cómputo, que la CFB. Al mismo tiempo, el problema inverso representa un caso interesante de estudio

para comparar las diversas estrategias evolutivas y genéticas, combinarlas, y eventualmente proponer algunas nuevas. Este es el objetivo de esta línea de investigación. Para obtener estos resultados se implementó un ambiente gráfico que permite testear y comparar los resultados del uso de dichas estrategias, y evaluar numérica y visualmente los resultados que se van produciendo a lo largo de las generaciones.

2 Sistemas de funciones iteradas

Un sistema de funciones iteradas (IFS) consiste en una colección de transformaciones afines contractivas que mapea al plano \mathbb{R}^2 sobre sí mismo. Esta colección de transformaciones define un mapa

$$W(\cdot) = \bigcup_{i=1}^n w_i(\cdot).$$

Las transformaciones afines contractivas tienen la importante característica que cuando son aplicadas repetidamente, convergen a un punto fijo. El mapa W no es aplicado a todo el plano sino a un subconjunto, es decir, a una colección de puntos del plano. Dado un conjunto inicial S , podemos calcular $w_i(S)$ para cada i , tomar la unión de estos conjuntos y obtener un nuevo conjunto $W(S)$. W es un mapa en el espacio de los subconjuntos del plano. Según el teorema de Hutchinson [2], cuando las transformaciones w_i son contractivas en el plano, entonces W es contractiva en el espacio (cerrado y acotado) de subconjuntos del plano. Por lo tanto, la iteración de un IFS converge a un único subconjunto del plano, el cual es un punto fijo del mapa W , denominado *atractor* A del IFS. Para la obtención del código IFS de una figura I arbitraria, se parte de una representación inicial aproximada de la silueta S del objeto que se desea modelar. La idea del método es buscar un grupo de mosaicos —copias transformadas de S — que en conjunto cubran la silueta original (Teorema del Collage [3]). Si por $d(I, J)$ denotamos la distancia de Hausdorff entre subconjuntos I y J del plano en una métrica adecuada, y r es la contractividad global de W , entonces el teorema del collage permite afirmar la desigualdad

$$d(A, I) \leq \frac{1}{1-r} d(I, \bigcup w_i(I)),$$

lo cual significa que la distancia entre la figura I y el atractor A del IFS está acotada por la distancia entre I y el cubrimiento determinado por W . A mejores cubrimientos, mejor aproximación a la imagen final. Como ya mencionáramos, este cubrimiento hasta ahora no ha podido implementarse en forma automática, debiéndose trabajar con asistencia humana. A partir de la silueta S , son introducidas una a una copias reducidas de S , de modo de poder ir cubriéndola con sus copias transformadas. Cada copia podrá ser torcida, escalada, rotada o trasladada según convenga para lograr el propósito buscado. Es importante que el tamaño de cada $w_n(S)$ sea menor que el de S para asegurar que cada transformación sea contractiva.

En otras palabras, para lograr la codificación se deben determinar un conjunto de transformaciones afines contractivas $\{w_1, w_2, \dots, w_n\}$ con la siguiente propiedad: La silueta original S y el conjunto

$$\tilde{S} = \bigcup_{n=1}^N w_n(S)$$

deben ser visualmente cercanos, siendo el número de transformaciones el menor posible. $w_n(S)$ es llamado el n -ésimo mosaico del collage. En la Figura 1 se muestra un atrac-

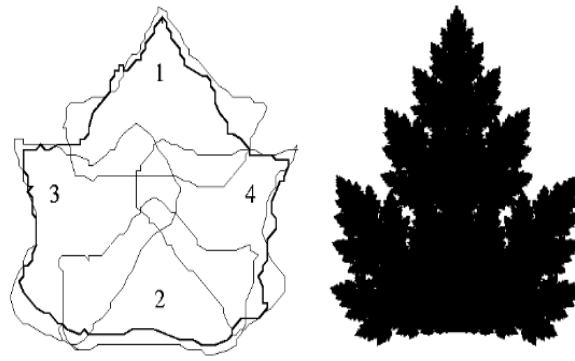


Figura 1: Cubrimiento de una hoja para la obtención de los códigos IFS y atractor que se logra con ellos.

tor obtenido utilizando el procedimiento descrito. Un algoritmo utilizado para generar el atractor o imagen a partir de las transformaciones, comienza con el código IFS $\{w_n, p_n : n = 1, 2, \dots, N\}$ junto con la ventana de graficación V de resolución $L \times M$. El algoritmo, denominado “juego del caos” [3], realiza una recorrida aleatoria del atractor del siguiente modo: parte de un punto inicial, escoge al azar un mapa w_n , y transforma el punto mediante este mapa, obteniendo las coordenadas de un pixel. Este pixel es graficado, y su punto correspondiente es luego sometido al mismo proceso, y así una cantidad predefinida de iteraciones.

3 Estrategias evolutivas y genéticas

La programación evolutiva (PE) y la programación genética (PG) son técnicas bien conocidas que surgen de imitar lo que conocemos de la evolución natural. La PE y la PG se utilizan, entre otros contextos, para resolver problemas que con los métodos de Inteligencia Artificial de alto nivel tradicionales (búsqueda heurística, lógica, etc.) son o bien intratables o bien imprácticos.

Las ideas principales provienen de una metáfora de la evolución natural, en la cual existen individuos (fenotipos) que expresan una información genética (genotipo), y además están sujetos a la presión evolutiva del medio (fitness). Los individuos más exitosos (con mejor fitness) ven aumentadas sus posibilidades reproductivas, generando nuevos individuos con su mismo genotipo. Sin embargo, en el proceso reproductivo ocurre el fenómeno distintivo de la evolución natural: la aparición de mutaciones al azar.

Este mismo esquema dinámico, con un sinnúmero de variaciones, ha sido desarrollado para diversas aplicaciones. En este trabajo veremos cómo el problema inverso de la IFS puede resolverse por medio de estrategias evolutivas y genéticas. Supongamos la existencia de una población con μ ancestros, cuyo genotipo \bar{g}_i se encuentra codificado con números. Entonces se crea un conjunto λ de descendientes siguiendo el esquema

$$\bar{p}_i = (\bar{g}_i, \bar{\sigma}_i) \xrightarrow{\text{mut}} (\bar{g}_i + n_0(\bar{\sigma}_i), \alpha(\bar{\sigma}_i)),$$

donde $n_0(\bar{\sigma}_i)$ denota la elección de un número de una aleatoria normal con media cero y variancia σ_i . α define una función de adaptación, la cual puede ser la distancia de Hamming entre el atractor A del IFS resultante y la imagen I de entrada.

En nuestro caso, \bar{g}_i representa los parámetros de la IFS W que estamos tratando de encontrar. Cada mapa w_k requiere seis reales para su objetivo:

$$w_k(x, y) = \begin{pmatrix} a_k & b_k \\ c_k & d_k \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e_k \\ f_k \end{pmatrix} \quad (1)$$

$$= \begin{pmatrix} r_{1k} \cos \theta_{1k} & -r_{2k} \sin \theta_{1k} \\ r_{1k} \sin \theta_{2k} & r_{2k} \cos \theta_{2k} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e_k \\ f_k \end{pmatrix} \quad (2)$$

En general, la expresión 2 es preferible a la 1 por tener un significado geométrico más intuitivo. Por lo tanto, \bar{g}_i representará las n sextuplas $(r_{1k}, r_{2k}, \theta_{1k}, \theta_{2k}, e_k, f_k)$, $k \in [1..n]$ en un determinado individuo de una generación. Como estrategia híbrida se puede introducir, además de la mutación, el *crossover* o cruzamiento entre el genotipo de dos ancestros para obtener el genotipo del sucesor.

Para evitar los mínimos locales, y quedar atrapado en un “nicho ecológico”, la estrategia mantiene una diversidad genotípica seleccionando un grupo numeroso de individuos con la mejor adaptación. Entre estas técnicas de *niching* podemos contar el *fitness sharing* (escalar el fitness en un entorno de cada individuo), el *crowding* (reemplazar al padre más cercano del individuo recientemente creado), y el *clustering* o agrupamiento, la cual resulta ser la más eficiente en la práctica.

4 Implementación de un ambiente para testear estrategias

Una vez definido el propósito y la metodología de trabajo, se implementó un prototipo en el lenguaje Matlab para testear el funcionamiento de los algoritmos y la eficacia de las estrategias. Luego, para contar con una implementación más adecuada en velocidad y capacidad de cómputo, se reescribió el programa en el lenguaje visual Delphi, construyéndose un ambiente gráfico para testear y evaluar las diferentes estrategias. En la Fig. 2 es posible observar la interfase gráfica del sistema. Se puede observar abajo a la izquierda el atractor resultante, y abajo a la derecha, la curva que muestra la evolución del *fitness* del atractor a lo largo de las generaciones. Los controles de la parte superior izquierda permiten elegir un conjunto de imágenes de entrada, para testear el correcto funcionamiento en varios casos, mientras que los controles a la derecha permiten controlar la ejecución del sistema, modificar los parámetros, las diversas estrategias, etc. La implementación en una herramienta como Delphi permitió migrar con cierto trabajo el prototipo original, pero con el beneficio de la velocidad de cómputo, la flexibilidad del uso de las componentes visuales, y otras posibilidades importantes. En la Fig. 2 se pueden ver los parámetros de la simulación realizada. Se realizaron 2000 generaciones, a 3000 órbitas por generación, con 3 transformaciones por genotipo, 10 antecesores y 5 sucesores. El tiempo total en un equipo PC standard fue de menos de dos minutos.

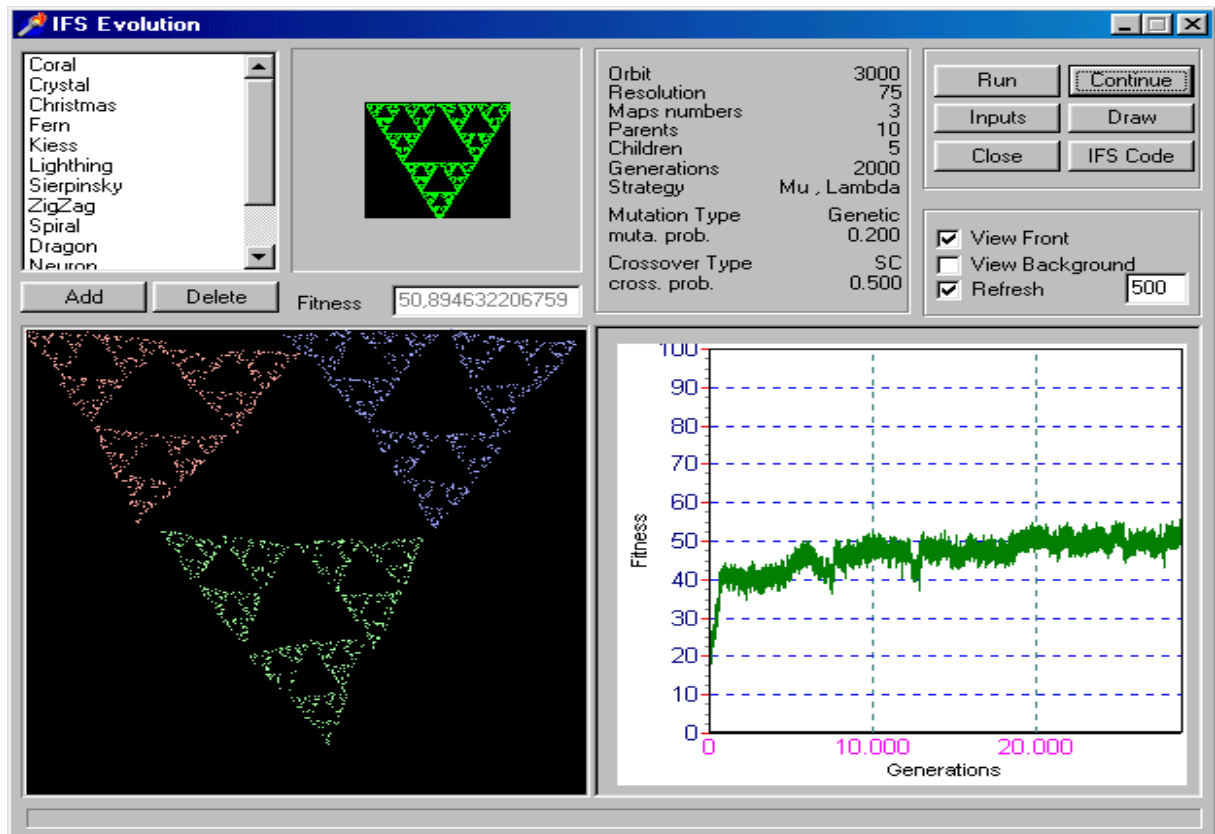


Figura 2: La interfase del sistema, mostrando la convergencia evolutiva de un conjunto de transformaciones al atractor de la imagen inicial.

Referencias

- [1] M. Barnsley y S. Demko. Iterated function systems and the global construction of fractals. *Proc. Roy. Soc. London*, A399:243–275, 1985.
- [2] M. Barnsley, A. Jacquin, y F. Malassenet. Harnessing Chaos for Image Synthesis. *ACM Computer Graphics*, 22(3):131–14, 1988.
- [3] M. F. Barnsley. *Fractals Everywhere*. Academic Press, Boston, 1988.
- [4] Yuval Fisher, editor. *Fractal Image Compression: Theory and Application*. Springer-Verlag, New York, NY, USA, 1995.
- [5] Yuval Fisher, editor. *Fractal Image Encoding and Analysis*, Springer-Verlag, New York, NY, USA, 1997.
- [6] Arnaud E. Jacquin. Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations. *SPIE Proceedings*, Vol. 1360, págs. 227–239, Lausanne, 1990.
- [7] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, segunda edición, 1994.
- [8] I. Rechenberg. *Evolution strategie: Optimierung technischer systeme nach prinzipien der biologischen evolution*. Frommann-Holzboog, 1973.