8. Computación Evolutiva

La computación evolutiva es un área de las ciencias de la computación que se enfoca al estudio de las propiedades de una serie de metaheurísticas estocásticas (a las cuales se les denomina "Algoritmos Evolutivos") inspiradas en la teoría de la evolución de las especies formulada por Charles Darwin, según la cual los individuos más aptos a su ambiente tienen una mayor probabilidad de sobrevivir.

Una metaheurística es un procedimiento de alto nivel que aplica una regla o conjunto de reglas que se basa(n) en una fuente de conocimiento. El énfasis de las metaheurísticas es explorar el espacio de búsqueda de manera relativamente eficiente; es decir, mejor que una búsqueda exhaustiva, aunque no necesariamente óptima. Las metaheurísticas suelen considerarse también como técnicas de búsqueda y optimización de uso general pese a sus restricciones teóricas.[1] Además, suelen requerir de poca información específica del problema y son útiles cuando el espacio de búsqueda es muy grande, poco conocido y/o con dificultades para explorarlo. Cuando se usan para optimización, no pueden garantizar, en general, que convergerán a la mejor solución posible (es decir, al óptimo global del problema), si bien en la práctica suelen producir aproximaciones razonablemente buenas en tiempos razonablemente cortos.

[1] Wolpert, D. H., Macready, W. G. (1997). **No Free Lunch Theorems for Optimization**. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.

Los Algoritmos Evolutivos (AEs) son metaheurísticas estocásticas porque su comportamiento se basa en el uso de números aleatorios y, por lo tanto, no necesariamente generan el mismo resultado cada vez que se ejecutan. Los AEs se han aplicado exitosamente a diversos tipos de problemas.[2] A lo largo de los años, se han desarrollado al menos dos tipos de AEs con base en el tipo de problemas que pretenden resolver: i) los diseñados para resolver problemas de optimización (sobre todo no lineal y combinatoria) y ii) los diseñados para resolver problemas de aprendizaje de máquina (sobre todo de clasificación). Sin embargo, la investigación en computación evolutiva no se ha limitado al desarrollo de nuevos algoritmos, sino también al diseño de nuevos operadores, mecanismos de selección y de representación, así como al modelado matemático de los AEs, el estudio de las fuentes de dificultad en un espacio de búsqueda y su correlación con los operadores de un AE y la hibridación de los AEs con otro tipo de técnicas.

El resto de este capítulo está organizado de la siguiente manera. Primero se revisan los antecedentes históricos de los AEs. Posteriormente, se describen los tres principales tipos de AEs que existen (las estrategias evolutivas, la programación evolutiva y los algoritmos genéticos), así como una variante muy popular de los algoritmos genéticos conocida como "programación genética". En dicha descripción se incluyen algunas aplicaciones relevantes, así como una breve discusión de trabajos de investigación que se han realizado en México. En la parte final del capítulo se mencionan otras metaheurísticas bio-inspiradas con las que actualmente se trabaja en nuestro país. También se proporciona una breve discusión sobre los orígenes de esta área de investigación en México y se concluye con una breve discusión de sus perspectivas.

8.1. Antecedentes históricos

La computación evolutiva tiene una historia larga y fascinante. En sus orígenes, varios investigadores propusieron de manera totalmente independiente

[2] Fogel, D. B. **Evolutionary Computation: Toward a New Philosophy of Machine Intelligence**, IEEE Press, Piscataway, NJ, USA, 1995.

diversos modelos computacionales con objetivos específicos, los cuales tenían en común su inspiración en la evolución natural. Con el tiempo, las evidentes similitudes entre estas técnicas llevaron a acuñar el término genérico "Algoritmo Evolutivo" en la década de los noventa del siglo pasado.[3] En esta sección, se discuten brevemente los acontecimientos más importantes de su historia. Cabe indicar que muchas de las bases de la computación evolutiva se desarrollaron en los sesenta y los setenta, cuando se propusieron los tres enfoques más populares dentro de la computación evolutiva: i) la programación evolutiva, ii) las estrategias evolutivas y iii) los algoritmos genéticos. Si bien los tres enfoques se inspiraron en las mismas ideas, difieren en sus detalles, como el esquema para representar soluciones y el tipo de operadores adoptados. Adicionalmente se discuten otras propuestas que se consideran importantes por su valor histórico pese a que su uso no llegó a popularizarse.

Wright[4] y Cannon[5] fueron los primeros investigadores que vieron a la evolución natural como un proceso de aprendizaje en el cual la información genética de las especies se cambia continuamente a través de un proceso de ensayo y error. De esta forma, la evolución puede verse como un método cuyo objetivo es maximizar la capacidad de adaptación de las especies en su ambiente. Wright fue más lejos e introdujo el concepto de "paisaje de aptitud" como una forma de representar el espacio de todos los genotipos posibles junto con sus valores de aptitud. Este concepto se ha utilizado extensamente en computación evolutiva para estudiar el desempeño de los AEs.[6] Adicionalmente, Wright también desarrolló uno de los primeros estudios que relacionan a la selección con la mutación, y concluyó que éstas deben tener

[3] Fogel, D. B. **Evolutionary Computation: The Fossil Record**, The Institute of Electrical and Electronic Engineers, New York, USA, 1998.

[4] S. Wright, S. (1932). **The roles of mutation, inbreeding, crossbreeding and selection in evolution**. En *VI International Congress of Genetics*, Vol. 1, pp. 356-366.

[5] Cannon, W. B., **The wisdom of the body**, W. W. Norton & Company, inc., 1932.

[6] Richter, H. (2014). **Fitness landscapes: From evolutionary biology to evolutionary computation**. En H. Richter and A. Engelbrecht (eds.), *Recent Advances in the Theory and Application of Fitness Landscapes, Emergence, Complexity and Computation*, Springer Berlin Heidelberg, Vol. 6, pp. 3–31.

un balance adecuado para tener éxito en el proceso evolutivo. Las ideas de Wright se extendieron por Campbell,[7] quien afirmaba que el proceso de variación ciega combinado con un proceso de selección adecuado constituye el principio fundamental de la evolución.

Turing puede ser considerado también uno de los pioneros de la computación evolutiva, pues reconoció una conexión (que a él le parecía obvia) entre la evolución y el aprendizaje de máquina.[8] Específicamente, Turing afirmaba que para poder desarrollar una computadora capaz de pasar la famosa prueba que lleva su nombre (con lo cual sería considerada "inteligente"), podría requerirse un esquema de aprendizaje basado en la evolución natural. Turing ya había sugerido antes que este tipo de métodos basados en la evolución podrían usarse para entrenar un conjunto de redes análogas a lo que hoy conocemos como "redes neuronales".[9] De hecho, Turing incluso usó el término "búsqueda genética" para referirse a este tipo de esquemas. Sin embargo, debido a que su supervisor consideró que este documento lucía más como un ensayo juvenil que como una propuesta científica,[10] el trabajo se publicó hasta 1968,[11] cuando existían ya algunos algoritmos evolutivos y las ideas de Turing al respecto acabaron por ser ignoradas.

Hacia finales de los cincuenta y principios de los sesenta se realizaron varios avances relevantes que coincidieron con el período en que las computadoras digitales se volvieron más accesibles. Durante dicho período, se

[7] Campbell, D. T. (1960). **Blind Variation and Selective Survival as a General Strategy in Knowledge-Processes**, En M. C. Yovits, S. Cameron (eds.), *Self-Organizing Systems*, Pergamon Press, New York, USA, pp. 205–231.

[8] Turing, A. M. (1950). **Computing Machinery and Intelligence**. *Mind,* 59:433–460.

[9] Turing, A. M. (1948). **Intelligent Machinery**. Report, National Physical Laboratory, Teddington, UK.

[10] Burgin M., Eberbach, E. (2013). **Recursively Generated Evolutionary Turing Machines and Evolutionary Automata**. En X.-S. Yang (ed.), *Artificial Intelligence, Evolutionary Computing and Metaheuristics*, Studies in Computational Intelligence, Vol. 427, Springer, Berlin, Germany, pp. 201–230.

[11] Evans, C. R., Robertson, A. D. J. (eds.) **Cybernetics: key papers**, University Park Press, Baltimore, Maryland, USA, 1968.

llevaron a cabo varios análisis de la dinámica poblacional que aparece durante la evolución. Una revisión del estado del arte en torno a este tema se presentó por Crosby en 1967,[12] quien identificó tres tipos de esquemas:

1. Métodos basados en el estudio de formulaciones matemáticas que emergen en el análisis determinista de la dinámica poblacional. Estos esquemas usualmente presuponen un tamaño de población infinito y analizan las distribuciones de los genes[13] bajo distintas circunstancias.

2. Enfoques que simulan la evolución pero que no representan explícitamente una población.[14] En estos métodos, las frecuencias de los diferentes genotipos[15] potenciales se almacenan para cada generación y resultan en un esquema no escalable en términos del número de genes. Adicionalmente, se requiere una representación matemática del sistema evolutivo.

3. Esquemas que simulan la evolución y que sí mantienen una representanción explícita de la población. Fraser[16] fue pionero en este tipo de métodos, los cuales se describen en una serie de artículos publicados a lo largo de una década.[17] Las conclusiones principales que se derivaron de dichos estudios se reúnen en un libro seminal sobre el tema.[18] Desde

[12] Crosby, J. L. (1967). **Computers in the study of evolution**, *Science Progress Oxford*, 55:279–292.

[13] Un "gene" es una secuencia de ácido desoxirribonucleico (ADN) que ocupa una posición específica en un cromosoma. A su vez, un "cromosoma" es una estructura dentro del núcleo de una célula que contiene el material genético que conforma a un individuo.

[14] Crosby, J. L. (1960). **The Use of Electronic Computation in the Study of Random Fluctuations in Rapidly Evolving Populations**. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 242(697):550–572.

[15] Se llama "genotipo" a la información genética de un organismo. Esta información (contenida en los cromosomas del organismo) puede o no ser manifestada u observada en el individuo.

[16] Fraser, A. S. (1957). **Simulation of genetic systems by automatic digital computers. I. Introduction**, *Australian Journal of Biological Science*, 10:484–491.

[17] Fraser, A. S., Burnell, D. (1967). **Simulation of Genetic Systems XII. Models of Inversion Polymorphism**. *Genetics*, 57:267–282.

[18] Fraser, A. S., Burnell, D. **Computer models in genetics**, McGraw-Hill, New York, USA, 1970.

su creación, este modelo se aceptó ampliamente y se adoptó por varios investigadores. De hecho, aunque existen varios detalles de implementación en los que las simulaciones de Fraser difieren de los algoritmos evolutivos modernos, algunos autores consideran al trabajo de Fraser como la primera propuesta de un algoritmo genético.[19] Cabe mencionar que aunque a nivel de la implementación la diferencia principal entre la propuesta de Fraser y el algoritmo genético moderno radica en que la primera usaba cromosomas diploides (es decir, un par de cromosomas por individuo) mientras que el segundo usa cromosomas haploides (es decir, un solo cromosoma por individuo), conceptualmente, el trabajo de Fraser se centró en analizar la dinámica poblacional y no en resolver problemas, que es la aplicación típica de los algoritmos evolutivos.

En este mismo período otros investigadores propusieron esquemas similares a un algoritmo evolutivo. De entre ellos, destacan Friedman, Box y Friedberg por el impacto que su trabajo tuvo posteriormente. Friedman[20] propuso el diseño automático de circuitos de control usando un esquema inspirado en la evolución natural, al que denominó "retroalimentación selectiva." Su propuesta era muy diferente a los algoritmos evolutivos modernos. Por ejemplo, no manejaba la noción de población[21] ni de generación.[22] Además, no implementó su método y planteó ciertas presuposiciones que resultaron demasiado optimistas. No obstante, su trabajo se considera pionero dentro de un área que hoy se conoce como "hardware evolutivo", en la que se busca diseñar circuitos con algoritmos evolutivos (en los que se adopta frecuentemente implementaciones en hardware).

---

[19] David B. Fogel, **Evolutionary Computation: The Fossil Record**, The Institute of Electrical and Electronic Engineers, New York, USA, 1998.

[20] Friedman, G, J., **Selective Feedback Computers for Engineering Synthesis and Nervous System Analogy**, MSc Dissertation, University of California, Los Angeles, 1956.

[21] La mayor parte de los algoritmos evolutivos modernos operan sobre un conjunto de soluciones potenciales al problema que están resolviendo. A dicho conjunto de soluciones se le denomina "población".

[22] Los algoritmos evolutivos se ejecutan durante un cierto número de iteraciones, a las cuales se les denomina "generaciones".

Box[23] propuso una técnica a la que denominó "Operación Evolutiva" (*Evolutionary Operation*, o EVOP) para optimizar el manejo de procesos de la industria química. EVOP usa una solución base (el "padre") para generar varias soluciones adicionales (los "hijos"), modificando unos cuantos parámetros de producción a la vez. Posteriormente, un individuo se selecciona para pasar a la siguiente generación, a partir de cierta evidencia estadística. Este sistema no era autónomo, pues requería de asistencia humana en los procesos de variación y selección. De tal forma, EVOP puede considerarse como el primer algoritmo evolutivo interactivo.

La contribución más importante de Friedberg[24] fue su intento de generar automáticamente programas de computadora mediante un esquema evolutivo. En sus primeros trabajos,[25] no identificó ninguna relación específica entre su propuesta y la evolución natural; sin embargo, dicha relación se identificó en publicaciones posteriores de sus co-autores.[26] En sus primeros intentos, el objetivo era generar automáticamente programas pequeños con algunas funcionalidades simples basadas en un conjunto de instrucciones hechas a la medida. Para poder dirigir la búsqueda a regiones prometedoras, se incorporaba información dependiente del problema, mediante la definción de un esquema de variación diseñado específicamente para el problema a resolverse. Pese al uso de operadores especializados, el éxito de esta técnica fue limitado. Sin embargo, algunos de los análisis realizados por Friedberg y sus co-autores fueron particularmente importantes para algunos de los logros posteriores de la computación evolutiva. Entre sus contribuciones más importantes, destacan las siguientes:

---

[23] Box, G. E. P. (1957). **Evolutionary Operation: A Method for Increasing Industrial Productivity**. *Applied Statistics*, 6(2):81–101.

[24] Friedberg, R. M. (1958). **A Learning Machine: Part I**. *IBM Journal of Research and Development*, 2(1):2–13.

[25] Friedberg, R. M., Dunham, B., North, J. H. (1959). **A Learning Machine: Part II**. *IBM Journal of Research and Development*, 3(3):282–287.

[26] Dunham, B., Fridshal, D., Fridshal, R., North, J. H. (1963). **Design by natural selection**. *Synthese*, 15(1): 254–259.

- La idea de dividir las soluciones candidatas en clases es precursora de las nociones de paralelismo implícito[27] y de esquema[28] que propusiera años después John Holland.[29]

- Se probaron varios conjuntos de instrucciones, mostrándose, por primera vez, la influencia del mapeo del genotipo al fenotipo.

- Se utilizó un algoritmo de asignación de crédito para medir la influencia de las instrucciones, de manera aislada. Esta idea está muy relacionada con el trabajo que realizara años después Holland con los algoritmos genéticos y los sistemas de clasificadores.

Algunos artículos no llamaron mucho la atención cuando se publicaron originalmente, pero propusieron técnicas que se reinventaron varios años después. Por ejemplo, el trabajo de Reed, Toombs y Barricelli[30] proporcionó varias innovaciones que se asemejan mucho a los conceptos de auto-adaptación,[31] cruza[32] y coevolución[33] que se usan hoy en día en computación evolutiva.

[27] El "paralelismo implícito" es una propiedad atribuída a los algoritmos genéticos, la cual les permite explorar más eficientemente el espacio de búsqueda mediante un proceso de estimación de la calidad de las soluciones.

[28] Un "esquema" es una abstracción usada para representar las soluciones que procesa un algoritmo genético. Holland usó este concepto para analizar matemáticamente la forma en la que opera un algoritmo genético.

[29] Holland, J. H. **Adaptation in Natural and Artificial Systems**, University of Michigan Press, Ann Arbor, Michigan, USA, 1975.

[30] Reed, J., Toombs, R., Barricelli, N. A. (1967). **Simulation of biological evolution and machine learning: I. Selection of self-reproducing numeric patterns by data processing machines, effects of hereditary control, mutation type and crossing**". *Journal of Theoretical Biology*, 17(3):319–342.

[31] La "auto-adaptación" es un mecanismo que permite que un algoritmo evolutivo adapte por sí mismo los valores de sus parámetros.

[32] La "cruza" es un operador utilizado en los algoritmos evolutivos para hacer que dos individuos intercambien material genético y produzcan descendientes (o "hijos") que combinen dicho material.

[33] La "coevolución" es un modelo en computación evolutiva en el cual se usan dos poblaciones que interactúan entre sí de tal forma que la aptitud de los miembros de una de ellas depende de la otra población y vice versa. En otras palabras, en este modelo, la evolución de una especie está condicionada a la de otra especie.

8.2. Programación Evolutiva

Este tipo de algoritmo evolutivo se propuso originalmente por Lawrence Fogel a principios de los sesenta cuando realizaba investigación básica en torno a inteligencia artificial para la *National Science Foundation* en Estados Unidos.[34] En esa época, la mayor parte de los intentos existentes para generar comportamientos inteligentes usaban al humano como modelo. Sin embargo, Fogel consideró que, puesto que la evolución fue capaz de crear a los humanos y a otras criaturas inteligentes su simulación podría conducir a comportamientos inteligentes.[35]

Las ideas básicas de Fogel eran similares a las de algunos trabajos que describimos anteriormente: usar mecanismos de selección y variación para evolucionar soluciones candidatas que se adaptaran mejor a las metas deseadas. Sin embargo, dado que Fogel desconocía dichos trabajos, su propuesta puede considerarse una re-invención de los esquemas basados en la evolución. Filosóficamente, las estructuras de codificación utilizadas en la programación evolutiva son una abstracción del fenotipo[36] de diferentes especies.[37] Consecuentemente, la codificación de soluciones candidatas se puede adaptar libremente para satisfacer los requerimientos del problema (es decir, no se requiere codificar las soluciones en un alfabeto específico como ocurre, por ejemplo, con el Algoritmo Genético, en donde las soluciones normalmente se codifican usando un alfabeto binario). Dado que no es posible la recombinación sexual entre especies diferentes, la Programación Evolutiva no incorpora un operador de cruza o recombinación. La ausencia del operador de cruza, la libertad de adaptar la codificación y el uso de un operador de selección pro-

[34] Fogel, L. J. (1962). **Autonomous automata**. *Industrial Research,* 4:14–19.

[35] Fogel, L. J. (1999). **Intelligence through simulated evolution: forty years of evolutionary programming**, Wiley series on intelligent systems, Wiley, 1999.

[36] El "fenotipo" es un término que se usa en genética para denotar los rasgos físicos visibles de un individuo. En algoritmos evolutivos, el fenotipo se refiere al valor decodificado de una solución candidata.

[37] Fogel, D. B. (1994). **An introduction to simulated evolutionary optimization**. *IEEE Transactions on Neural Networks*, 5(1):3–14.

babilístico[38] (el cual no se incorporó en las primeras versiones del algoritmo) son los rasgos principales que distinguen a la programación evolutiva de los demás algoritmos evolutivos.

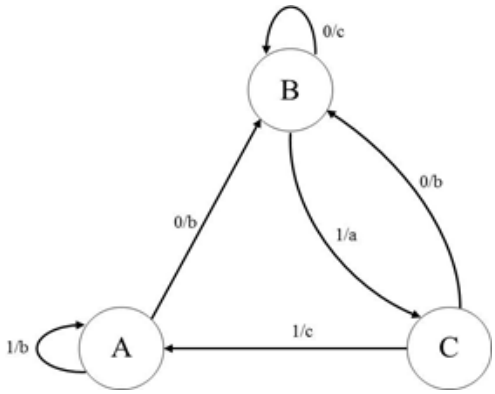

Figura 8.1 Máquina de estados finitos como las usadas por Lawrence Fogel en sus primeros experimentos con la Programación Evolutiva. Los símbolos a la izquierda y a la derecha del "/" son de entrada y de salida respectivamente. El estado inicial en este caso es el C.

En sus primeras versiones de la Programación Evolutiva, Fogel se percató de que uno de los principales rasgos que caracterizan el comportamiento inteligente es la capacidad de predecir el ambiente, junto con un mecanismo que permita traducir las predicciones a una respuesta adecuada de manera que se logre el objetivo deseado. Por otra parte, un transductor de estados finitos es una máquina que permite transformar una secuencia de símbolos de entrada en una secuencia de símbolos de salida, como la que se ilustra en la Figura 8.1. Dependiendo de los estados y transiciones, esta máquina se puede usar para modelar diferentes situaciones y producir diferentes transformaciones. Por tanto, Fogel la consideró como un mecanismo adecuado para lidiar con el problema de generar comportamiento inteligente. En sus primeros experimentos, su objetivo fue desarrollar una máquina capaz de predecir el comportamiento de un ambiente, el cual se representaba como una secuencia

[38] El uso de un operador de selección probabilístico implica que los individuos menos aptos de la población tienen una probabilidad distinta de cero de ser seleccionados.

de símbolos de un alfabeto de entrada; es decir, dados los símbolos generados previamente por el ambiente, predecir el siguiente símbolo que emergerá del ambiente. Pensemos, por ejemplo, que queremos obtener un transductor de estados finitos como el que se muestra en la figura 8.1. Para este efecto se requiere crear una tabla de transiciones como la que se muestra en la tabla 8.1. En ésta se indica, para cada estado, el siguiente estado según la entrada, y el símbolo de salida. Fogel intentó generar estas máquinas de manera automatizada (simulando la evolución), usando como entrada los símbolos de salida de la tabla de transiciones. El objetivo era que la Programación Evolutiva produjera una máquina de estados finitos que generara los símbolos de salida deseados, sin intervención humana.

Estado Actual	C	B	C	A	A	B
Símbolo de entrada	0	1	1	1	0	1
Estado siguiente	B	C	A	A	B	C
Símbolo de salida	b	a	c	b	b	a

Tabla 8.1 Esta tabla ilustra las transiciones de la máquina de estados finitos que se muestra en la figura 8.1.

El algoritmo original de Programación Evolutiva funcionaba de la manera siguiente: Primero, se producía de manera aleatoria una población con *N* máquinas de estados finitos. Posteriormente, cada miembro de la población era mutado, creándose una población de hijos del mismo tamaño de la de los padres. Fogel adoptó cinco tipos de operadores de mutación: i) agregar un estado, ii) borrar un estado, iii) cambiar el estado siguiente de una transición, iv) cambiar el símbolo de salida de una transición, o v) cambiar el estado inicial. Los operadores de mutación se elegían aleatoriamente[39] y, en algunos casos, los hijos se mutaban más de una vez. Finalmente, los hijos eran evaluados y se seleccionaban las *N* mejores máquinas (de entre los padres e hijos). Cada máquina de estados finitos era evaluada en términos de su capacidad para

[39] Fogel experimentó con diversos esquemas para elegir el operador de mutación más adecuado, pero al final decidió adoptar un método de selección aleatorio.

predecir correctamente los siguientes símbolos en las secuencias conocidas. Inicialmente, la aptitud[40] se calculaba considerando un número pequeño de símbolos, pero conforme la evolución progresaba, se agregaban más símbolos al conjunto de entrenamiento. En sus primeros experimentos Fogel realizó pruebas con diferentes tipos de tareas de predicción: secuencias periódicas de números, secuencias con ruido, ambientes no estacionarios, etc. Posteriormente, consideró tareas más complejas como el reconocimiento de patrones, la clasificación y el diseño de sistemas de control. Todo este trabajo originó el que se considera el primer libro de computación evolutiva de la historia.[41] Este trabajo fue muy importante porque presenta, entre otras cosas, una de las primeras aplicaciones de la coevolución. También es importante enfatizar que en la mayor parte de los primeros estudios en torno a la Programación Evolutiva, los resultados computacionales presentados no eran muy amplios, debido al limitado poder de cómputo de la época. La mayor parte de estos experimentos iniciales fueron recapitulados y extendidos varios años después, proporcionando una mayor comprensión de los verdaderos alcances de la Programación Evolutiva.[42]

En los años setenta la mayor parte de la investigación en torno a la Programación Evolutiva se realizó bajo la tutela de Dearholt. Una de las principales contribuciones de su trabajo fue la aplicación de Programación Evolutiva a problemas prácticos. Por ejemplo, al reconocimiento de patrones en expresiones regulares[43] y en caracteres manuscritos,[44] así como para clasificar

[40] La "aptitud" de un individuo es una medida que permite comparar una solución con respecto a las demás. Si el problema a resolverse es de optimización, normalmente la aptitud se define en términos de la función objetivo que se desea optimizar.

[41] Fogel, L. J., Owens, A. J., Walsh, M. J. **Artificial Intelligence Through Simulated Evolution**, John Wiley & Sons, USA, 1966.

[42] Fogel, L. J., Fogel, D. B. **Artificial intelligence through evolutionary programming**, Technical Report PO-9-X56-1102C-1, U.S. Army Research Institute, San Diego, California, USA, 1986.

[43] Lyle, M. R. **An Investigation Into Scoring Techniques in Evolutionary Programming**, MSc Thesis, Las Cruces, USA, 1972.

[44] Cornett, F. N. **An Application of Evolutionary Programming to Pattern Recognition**, MSc Thesis, Las Cruces, USA, 1972.

diferentes tipos de electrocardiogramas.[45] Estos trabajos incorporaron varias novedades algorítmicas. De entre ellas, tal vez las más importantes hayan sido el uso simultáneo de varios operadores de mutación y la adaptación dinámica de las probabilidades asociadas con los diferentes esquemas de mutación.

A principios de los ochenta, la Programación Evolutiva se diversificó usando otras representaciones arbitrarias de las soluciones candidatas a fin de resolver diferentes tipos de problemas. Eventualmente se utilizó, por ejemplo, para resolver problemas de ruteo mediante una codificación basada en el uso de permutaciones de enteros,[46] y para generar programas de computadora de manera automática, adoptando una codificación de árbol.[47] Asimismo, se usaron vectores de números reales para resolver problemas de optimización continua[48] y para entrenar redes neuronales.[49] Durante este período, se aceptaba comúnmente que el diseñar representaciones específicas para un problema, junto con operadores especializados, permitía realizar una búsqueda más eficiente.[50] De hecho, varios investigadores defendían la posición de que el diseño de esquemas de mutación inteligentes podía evitar el uso de los operadores de recombinación.[51]

En los noventa, el uso de la Programación Evolutiva en problemas de optimización continua se extendió considerablemente. Durante este perío-

[45] Dearholt, D. W. (1976). **Some experiments on generalization using evolving automata**. En *9th Hawaii International Conference on System Sciences*, Western Periodicals, Honolulu, Hawaii, USA, pp. 131–133.

[46] Ver nota 42.

[47] Chellapilla, K. (1997). **Evolving Computer Programs Without Subtree Crossover**. *IEEE Transactions on Evolutionary Computation*, 1(3):209–216.

[48] Yao, X., Liu, Y., Lin, G. (1999). **Evolutionary programming made faster**. *IEEE Transactions on Evolutionary Computation*, 3(2):82–102.

[49] Porto, V. W., Fogel, D. B. (1995). **Alternative neural network training methods [active sonar processing]**. *IEEE Expert*, 10(3):16–22.

[50] Fogel, L. J. **Intelligence through simulated evolution: forty years of evolutionary programming**, Wiley series on intelligent systems, Wiley, USA, 1999.

[51] Fogel, D. B., Atmar, J. W. (1990). **Comparing genetic operators with gaussian mutations in simulated evolutionary processes using linear systems**. *Biological Cybernetics*, 63(2):111–114.

do se hicieron enormes esfuerzos por desarrollar esquemas auto-adaptativos. Cuando se usa auto-adaptación, algunos parámetros del algoritmo se codifican junto con las variables del problema, de tal manera que los operadores evolutivos actúen sobre ellos y definan sus valores. Bajo este tipo de esquema, las variables del problema son llamadas parámetros objeto, mientras que los parámetros del algoritmo evolutivo se denominan parámetros de la estrategia. En los primeros intentos por incorporar auto-adaptación a la Programación Evolutiva, se adaptaba el factor de escalamiento de la mutación Gaussiana.[52] En otras variantes más avanzadas se consideran varios operadores de mutación simultáneamente.[53] Puesto que los esquemas auto-adaptativos resultaron tener un desempeño superior a las variantes tradicionales, este mecanismo se volvió muy popular en muchas de las implementaciones usadas para resolver problemas del mundo real, particularmente cuando se adoptaba codificación mediante vectores de números reales.[54] Cabe destacar que el mecanismo de auto-adaptación de la Programación Evolutiva para optimización continua presenta varias similitudes con respecto al mecanismo adoptado en las Estrategias Evolutivas. La diferencia más significativa es que la Programación Evolutiva no incorpora recombinación. Adicionalmente, algunos detalles de implementación eran también diferentes en las primeras variantes auto-adaptativas de la Programación Evolutiva.[55] Por ejemplo, el orden en el que se sometían a mutación los parámetros objeto y los de la estrategia era diferente en la Programación Evolutiva y las Estrategias Evolutivas, si bien estas diferencias acabaron por desvanecerse con el tiempo. Además, existen casos documentados en los que se ha podido mostrar que el uso exclusivo del operador de mutación produce mejores resultados que cuando se incorpora recombinación y viceversa.[56] Hay evidencia que indica que es benéfico adap-

[52] Fogel, D. B., Fogel, L. J., Atmar, J. W. (1991). **Meta-Evolutionary Programming**. En *Asilomar Conference in Signals Systems and Computers*, pp. 540–545.

[53] Ver nota 48.

[54] Fogel, D. B. **Evolutionary Computation: Toward a New Philosophy of Machine Intelligence**, IEEE Press, Piscataway, New Jersey, USA, 1995.

[55] Eiben, A. E., Smith, J. E. **Introduction to Evolutionary Computing**, Natural Computing Series, Springer, 2003.

[56] Richter, J. N., Wright, A., Paxton, J. **Ignoble Trails - Where Crossover Is Provably**

tar también los parámetros asociados con el operador de cruza usado por las Estrategias Evolutivas,[57] lo cual hace que disminuyan las diferencias entre la Programación y las Estrategias Evolutivas. Se hace notar, sin embargo, que el primer mecanismo de auto-adaptación de parámetros para optimización continua se produjo en las Estrategias Evolutivas, por lo cual este tema se discutirá a mayor detalle cuando hablemos de dicho tipo de algoritmo evolutivo.

En la actualidad, la Programación Evolutiva es el tipo de algoritmo evolutivo menos usado a nivel mundial. Aunque su uso original fue en problemas de predicción hoy en día se suele usar más para optimización en espacios continuos. Esto se refleja también en México, en donde existe muy poca evidencia de su uso para optimización mono-objetivo[58] y multi-objetivo[59] en espacios continuos.

8.3. Estrategias Evolutivas

A mediados de la década de los sesenta, tres estudiantes de la Universidad Técnica de Berlín (Bienert, Schwefel y Rechenberg) estudiaban problemas prácticos de mecánica de fluídos y otras áreas similares con el fin de construir robots que pudieran resolver de manera automatizada problemas de

---

Harmful. En G. Rudolph, T. Jansen, S. Lucas, C. Poloni, and N. Beume (eds.), *Parallel Problem Solving from Nature PPSN X*, pp. 92-101, Lecture Notes in Computer Science, Vol. 5199, Springer, Berlin, Germany, 2008.

[57] Jain, A. Fogel, D. B. (2000). **Case studies in applying fitness distributions in evolutionary algorithms II: Comparing the improvements from crossover and Gaussian mutation on simple neural networks**. En *2000 IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks*, pp. 91–97.

[58] Coello Coello, C. A., Landa Becerra, R. (2002). **Adding Knowledge and Efficient Data Structures to Evolutionary Programming: A Cultural Algorithm for Constrained Optimization**". En W.B. Langdon et al. (Editors), *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2002*, Morgan Kaufmann Publishers, San Francisco, California, USA, pp. 201–209.

[59] Coello Coello, C. A., Landa Becerra, R. (2003). **Evolutionary Multiobjective Optimization using a Cultural Algorithm**. En *2003 IEEE Swarm Intelligence Symposium*, IEEE Service Center, Indianapolis, Indiana, USA, pp. 6–13,

ingeniería.[60] Para este efecto, formularon a dichas tareas como problemas de optimización y desarrollaron máquinas autónomas que se modificaban automáticamente bajo la aplicación de ciertas reglas. También intentaron aplicar algoritmos clásicos de optimización. Sin embargo, debido a que los problemas que querían resolver tenían ruido y eran multimodales, estos algoritmos fueron incapaces de producir soluciones aceptables. Para enfrentar esta problemática decidieron aplicar métodos de mutación y selección análogos a la evolución natural. Rechenberg publicó el primer reporte en torno a la nueva técnica, denominada "Estrategia Evolutiva", aplicada a la minimización del arrastre total de un cuerpo en un túnel de viento.[61] Posteriormente, resolvieron otros problemas tales como el diseño de tubos y de boquillas hidrodinámicas.[62]

Filosóficamente, la codificación de soluciones adoptada por las Estrategias Evolutivas es una abstracción del fenotipo de los individuos.[63] Por esta razón se puede adoptar libremente la codificación de soluciones candidatas, a fin de adaptarse mejor a los requerimientos del problema. Además, en las Estrategias Evolutivas se permite la recombinación de individuos, si bien este operador no fue implementado en la versión original del algoritmo. A la primera versión de este algoritmo se le conoce como la (1+1)-EE (EE significa "Estrategia Evolutiva"). Su funcionamiento es el siguiente: primero, se crea una solución inicial de manera aleatoria y se considera como "el padre". Esta solución es mutada (es decir, el valor de una de sus variables es modificado aleatoriamente) y al individuo mutado se le denomina "el hijo". La mutación se aplicaba siguiendo una distribución binomial, a fin de aplicar mutaciones pequeñas más frecuentemente que las mutaciones grandes, como ocurre en la naturaleza. La solución mutada se vuelve el padre si es mejor o igual que el

[60] Fogel, D. B. **Evolutionary Computation: The Fossil Record**, Wiley-IEEE Press, USA, 1998.

[61] Rechenberg, I. **Cybernetic solution path of an experimental problem**, Technical Report, Library Translation 1122, Royal Air Force Establishment, 1965.

[62] Ver nota 60.

[63] Fogel, D. B. (1994). **An introduction to simulated evolutionary optimization**. *IEEE Transactions on Neural Network*s, Vol. 5, No. 1, pp. 3–14.

individuo que la originó. El nuevo padre se vuelve a mutar, repitiéndose este proceso hasta alcanzar un cierto criterio de paro. Los experimentos realizados por los creadores de esta técnica mostraron que resultaba mejor que los métodos de optimización clásicos en problemas de alta complejidad.

En 1965, Hans-Paul Schwefel implementó por primera vez una Estrategia Evolutiva en una computadora de uso general y la aplicó a problemas de optimización continua.[64] La contribución más importante de Schwefel fue que incorporó una mutación Gaussiana con media cero, y este operador se sigue usando a la fecha. El operador de mutación se podía controlar mediante el ajuste de las desviaciones estándar (es decir, el tamaño de paso) de la distribución Gaussiana. Los estudios realizados por Schwefel en torno al tamaño de paso originaron los primeros análisis teóricos de la Estrategia Evolutiva.

Hacia finales de los sesenta se introdujo una versión de la Estrategia Evolutiva que usaba una población con varios individuos. Esta primera versión poblacional fue denominada $(\mu+1)$-EE, y adoptaba μ individuos, los cuales se usaban para crear un nuevo individuo por medio de operadores de recombinación y mutación. Posteriormente, los μ mejores individuos de entre los $\mu+1$ existentes se seleccionaban para sobrevivir. Este esquema es muy similar a la selección de estado uniforme[65] que se volvió popular años más tarde en los algoritmos genéticos. Con el tiempo, las Estrategias Evolutivas permitieron adoptar cualquier número de padres y de hijos. Además, se comenzaron a utilizar dos esquemas de selección. Así surgió la $(\mu+\lambda)$-EE y la (μ,λ)-EE. En la primera, μ padres generan λ hijos y sobreviven los μ mejores de la unión de ellos (padres e hijos). En la segunda, μ padres generan λ hijos y sobreviven los μ mejores hijos.

---

[64] Schwefel, H. P. **Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik**, Dipl.-Ing. Thesis, Technical University of Berlin, Hermann Föttinger, Institute for Hydrodynamics, 1965.

[65] Whitley D., J. Kauth, J. **GENITOR: A different genetic algorithm**, Technical Report, ColoradoState University, Department of Computer Science, 1988.

Las primeras publicaciones sobre las Estrategias Evolutivas fueron escritas en alemán, lo cual limitó su acceso. Sin embargo, en 1981, Schwefel publicó el primer libro sobre estas estrategias en inglés.[66] Desde entonces, un número importante de investigadores en el mundo las han estudiado y aplicado extensamente. En México, se han utilizado para resolver problemas de procesamiento de imágenes[67] y para optimización no lineal con restricciones.[68]

El libro de Schwefel se enfoca en el uso de las Estrategias Evolutivas para optimización continua, que es de hecho el área en la que esta técnica ha tenido mayor éxito. En éste se discute, entre otros temas, el uso de diferentes tipos de operadores de recombinación y la adaptación de las distribuciones Gaussianas utilizadas para el operador de mutación.

En los estudios iniciales del operador de recombinación se propusieron cuatro esquemas posibles, combinando dos propiedades diferentes. Primero, se proporcionaron dos opciones para el número de padres involucrados en la creación de un hijo: i) recombinación bisexual (o local) y ii) recombinación multisexual (o global). En la primera se seleccionan dos padres y éstos se recombinan entre sí, produciendo así la solución candidata del hijo. En la segunda, se seleccionan diferentes padres para ir generando cada variable de la solución candidata del hijo. También hay dos opciones para combinar los valores de los padres: la denominada "recombinación discreta", en la que uno de los valores se selecciona aleatoriamente, y la denominada "recombinación

[66] Schwefel, H. P. **Numerical Optimization of Computer Models**, John Wiley & Sons, Inc., New York, New York, USA, 1981.

[67] Gómez García, H. F., González Vega, A., Hernández Aguirre, A., Marroquín Zaleta, J. L., Coello Coello, C. A. (2002). **Robust Multiscale Affine 2D-Image Registration through Evolutionary Strategies**, en Juan Julián Merelo Guervós, Panagiotis Adamidis, Hans-Georg Beyer, José-Luis Fernández-Villacañas y Hans-Paul Schwefel (editors), *Parallel Problem Solving from Nature VII*, Lecture Notes in Computer Science Vol. 2439, Springer-Verlag, Granada, Spain, September, pp. 740–748.

[68] Mezura Montes, E., Coello Coello, C. A. (2005). **A Simple Multi-Membered Evolution Strategy to Solve Constrained Optimization Problems**. *IEEE Transactions on Evolutionary Computation*, 9(1):1–17.

intermedia", en la que se calcula el promedio de los valores de los padres. En estos esquemas de recombinación no se puede especificar la cantidad de padres que toman parte en cada recombinación. Rechenberg[69] propuso una generalización que cambia esta restricción. En este caso, se usa un nuevo parámetro, denominado ρ para especificar el número de padres que tomarán parte en la creación de cada individuo. Esto originó las variantes denominadas (μ/ρ, λ)-EE y (μ/ρ+ λ)-EE. Posteriormente, se propusieron otras versiones que asignaban pesos a la contribución de cada uno de los individuos que participaban en la recombinación.[70]

Pese a utilizar recombinación, el operador principal de las Estrategias Evolutivas es la mutación. Como se indicó anteriormente, este operador se suele basar en una distribución Gaussiana con media cero. En la mayor parte de los casos, los individuos se perturban mediante la adición de un vector aleatorio generado de una distribución Gaussiana multivariada. En otras palabras, se usa la expresión siguiente:

$$x_i' = x_i + N_i\,(0,\, C)$$

donde x_i' es el nuevo individuo, generado a partir de x_i, C representa una matriz de covarianza y $N_i\,(0,\, C)$ devuelve un número aleatorio con media cero. De tal forma, el nuevo individuo (es decir, el "hijo") se crea a partir del anterior (el "padre") mediante una perturbación aleatoria a una de sus variables. Desde los orígenes de las Estrategias Evolutivas, resultó claro que C podría tener un efecto significativo en el desempeño del algoritmo. Consecuentemente, se han desarrollado diversos métodos para adaptar C durante la ejecución del algoritmo. Los primeros esquemas de este tipo se originaron de los estudios realizados por Rechenberg,[71] quien analizó las propiedades de

[69] Rechenberg, I. (1978). **Evolutionsstrategin**. En B. Schneider and U. Ranft (eds.), *Simulationsmethoden in der Medizin und Biologie*, Springer-Verlag, Berlin, Germany, pp. 83–114.

[70] Bäck, T., Schwefel, H. P. (1993). **An Overview of Evolutionary Algorithms for Parameter Optimization**. *Evolutionary Computation*, 1(1):1–23.

[71] Rechenberg, I. **Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution**, Frommann-Holzboog, Stutgart, Germany, 1973.

convergencia de las Estrategias Evolutivas en relación a la frecuencia relativa de las mutaciones exitosas (si el hijo es mejor que su padre en términos del valor de la función objetivo). Este trabajo condujo al primer esquema adaptativo en el cual el tamaño de paso global se ajusta mediante un procedimiento en línea con el objetivo de producir mutaciones exitosas con una tasa igual a 1/5. Este procedimiento se conoce como la "regla de éxito 1/5". Esta regla hace ajustes a la desviación estándar cuando la tasa de éxito no es exactamente 1/5, bajo la premisa de que esta tasa de mutaciones exitosas es la ideal. Es importante hacer notar que dicha regla no es general, lo cual originó las versiones auto-adaptativas de las Estrategias Evolutivas. En el primer esquema de este tipo, el único parámetro adaptado fue el tamaño de paso global. Sin embargo, con los años, se desarrollaron esquemas auto-adaptativos mucho más sofisticados.

Otro punto interesante en torno a las Estrategias Evolutivas es que si se adapta toda la matriz de covarianza pueden inducirse correlaciones entre las mutaciones realizadas en las cuales se involucran diferentes parámetros, lo cual hace más apropiado su uso para lidiar con funciones no separables. El primer esquema en el cual se adapta toda la matriz de covarianza fue propuesto por Schwefel.[72] En este caso, el sistema coordenado en el cual se realiza el control del tamaño de paso, así como los tamaños de paso mismos, se auto-adaptan. Se hace notar, sin embargo, que dicho esquema no ha sido muy exitoso porque requiere de tamaños de población muy grandes para operar adecuadamente. Una alternativa más eficiente es la Estrategia Evolutiva con un esquema de adaptación de la matriz de covarianza propuesta por Hansen y Ostermeier conocida como CMA-ES.[73] En este esquema se combinan dos técnicas diferentes de auto-adaptación con el objetivo de construir matrices de covarianza que maximicen la creación de vectores de mutación que fueron exitosos en generaciones anteriores. Esta técnica es muy compleja y requiere

[72] Schwefel, H. P., **Numerical Optimization of Computer Models**, John Wiley & Sons, Inc., New York, New York, USA, 1981.

[73] Hansen, N., Ostermeier, A. (2001). **Completely Derandomized Self-Adaptation in Evolution Strategies**. *Evolutionary Computation*, 9(2):159–195.

de varios parámetros, lo cual ha dado pie a variantes más simples y con menos parámetros.[74]

Finalmente, existen también esquemas que favorecen ciertas direcciones de búsqueda sin adaptar toda la matriz de covarianza. Por ejemplo, Poland y Zell[75] propusieron utilizar la dirección principal de descenso para guiar la búsqueda. La mayor ventaja de este tipo de esquemas es que reducen la complejidad computacional en términos de tiempo y espacio. Sin embargo, dado que el tiempo para calcular las matrices de covarianza no es significativo con respecto al de la evaluación de la función objetivo (sobre todo, si el problema es costoso en términos de tiempo de cómputo) este tipo de esquemas no ha sido muy popular, aunque puede resultar útil en problemas con un gran número de variables, pues en ese caso, el costo de calcular matrices de covarianza se incrementa de manera significativa.

Las Estrategias Evolutivas son una opción muy buena para resolver problemas de optimización continuos en los que todas las variables son números reales, aunque en años recientes han sido desplazadas por metaheurísticas como la evolución diferencial, que resultan más fáciles de implementar y usar.

8.4. Algoritmos Genéticos

John Holland[76] identificó la relación entre el proceso de adaptación que existe en la evolución natural y la optimización, y conjeturó que dicho proceso podría jugar un papel crítico en otras áreas tales como el aprendizaje, el control

[74] Beyer, H. G., Sendhoff, B. (2008). **Covariance Matrix Adaptation Revisited – The CMSA Evolution Strategy**. En G. Rudolph, T. Jansen, S. Lucas, C. Poloni, and N. Beume (eds.), *Parallel Problem Solving from Nature - PPSN X*, pp. 123–132, Lecture Notes in Computer Science, Vol. 5199, Springer, Berlin, Germany.

[75] Poland, J. Zell, A. (2001). **Main Vector Adaptation: A CMA Variant with Linear Time and Space Complexity**. En L. Spector et al. (eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*, Morgan Kaufmann, San Francisco, California, USA, pp. 1050–1055.

[76] Holland, J. H. **Adaptation in Natural and Artificial Systems**, University of Michigan Press, Ann Arbor, Michigan, USA, 1975.

automático o las matemáticas. Su objetivo inicial era estudiar de manera formal la adaptación y propuso un entorno algorítmico inspirado en estas ideas, al cual denominó originalmente "planes reproductivos y adaptativos", pero que después renombró como "Algoritmo Genético". El funcionamiento de estos algoritmos se basa en la modificación progresiva de un conjunto de estructuras de datos con el objetivo de adaptarlas a un cierto ambiente. La forma específica de realizar dichas adaptaciones está inspirada en la genética y en la evolución natural. No obstante, en poco tiempo, los Algoritmos Genéticos se volvieron solucionadores de problemas en áreas tales como el aprendizaje de máquina[77] y el control automático. Sin embargo, su uso más extendido ha sido en optimización, tanto en espacios discretos como continuos.

Filosóficamente, la codificación utilizada en los Algoritmos Genéticos es una abstracción del genotipo de los individuos. En este caso, sin importar el tipo de variables que tenga el problema, éstas se deben codificar en binario. Aunque hoy en día es posible usar otras codificaciones, Holland argumentó que el alfabeto binario es universal y presenta varias ventajas, tanto de tipo práctico como teórico. Sin embargo, pese a sus ventajas tiene la desventaja evidente de que se requiere de un proceso de decodificación para convertir la cadena binaria en la variable que codifica. Asimismo, el uso de la codificación binaria puede en algunos casos introducir problemas debido a la discretización de un espacio de búsqueda que originalmente no era discreto (por ejemplo, cuando las variables son números reales). Tales problemas van desde usar una precisión inadecuada hasta el manejo de cadenas binarias de gran tamaño con las cuales es más ineficiente operar dentro del Algoritmo Genético.

A cada cadena binaria que codifica una variable se le denomina *gene*, y cada bit dentro de un gene se conoce como *alelo*. A la concatenación de todos los genes (es decir, al conjunto de todas las variables del problema) se le denomina *cromosoma*, como se ilustra en la Figura 8.2. Un individuo está for-

[77] Los Algoritmos Genéticos dieron pie a los **Sistemas de Clasificadores,** en los cuales se evolucionaron conjuntos de reglas del tipo IF-THE-ELSE para resolver problemas de clasificación.

mado normalmente por un cromosoma (es decir, se considera una estructura haploide, pese a que en la naturaleza es muy común que las especies tengan una estructura diploide, es decir, con dos cromosomas).

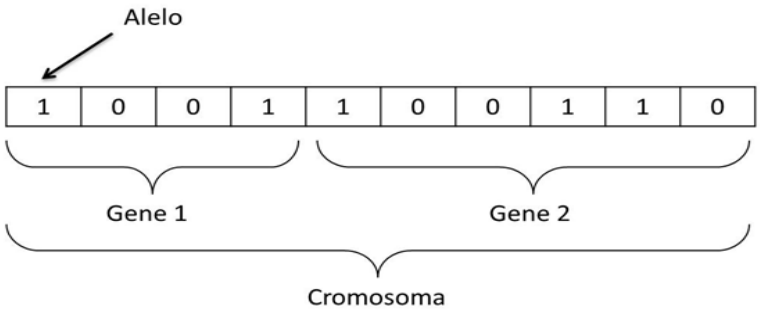

Figura 8.2 Estructura de una cadena cromosómica para un problema con dos variables ("gene 1" y "gene 2" corresponden a dichas variables).

La forma de operar de un Algoritmo Genético es bastante simple. Se parte de un conjunto de soluciones que se generan aleatoriamente al que se conoce como la "población inicial". Para cada una de estas soluciones (o "individuos") se debe calcular un valor de aptitud. Dicho valor está en función del objetivo que se quiere optimizar, aunque por lo general, requiere algún proceso de normalización a fin de evitar valores de aptitud negativos o divisiones entre cero. Las parejas de padres que se reproducirán se seleccionan de acuerdo a su valor de aptitud (los mayores de toda la población). En la versión original del Algoritmo Genético se utiliza un esquema de selección proporcional, lo que significa que los individuos más aptos tienen mayor probabilidad de sobrevivir. Sin embargo, los menos aptos tienen una probabilidad distinta de cero de ser seleccionados y podrían volverse padres también. Los individuos seleccionados se recombinan mediante un operador de cruza para generar la población de hijos, a la cual se aplica el operador de mutación. Los hijos mutados conforman la nueva población que reemplaza por completo a la población anterior. El proceso se repite hasta cumplirse un cierto criterio de paro, que normalmente es un número máximo de gene-

raciones (o de iteraciones). La tabla 8.2 muestra una población hipotética de cinco individuos de un algoritmo genético, incluyendo los cromosomas y las aptitudes correspondientes.

Individuo No.	Cromosoma	Aptitud	% del Total
1	11010110	254	24.5
2	10100111	47	4.5
3	00110110	457	44.1
4	01110010	194	18.7
5	11110010	85	8.2
Total		1037	100.0

Tabla 8.2 Ejemplo de una población de cinco individuos, cuyos cromosomas contienen 8 bits cada uno. En la tercera columna se muestra la aptitud (calculada mediante una función hipotética).

La figura 8.3 ilustra el funcionamiento de un método de selección proporcional conocido como "la ruleta", que se usa con los algoritmos genéticos. El triángulo negro es un pivote. En la selección por el método de la ruleta, se simula el giro aleatorio de un disco que contiene a los individuos de la población y en cada giro se selecciona como padre al individuo que marca el pivote al detenerse la ruleta.

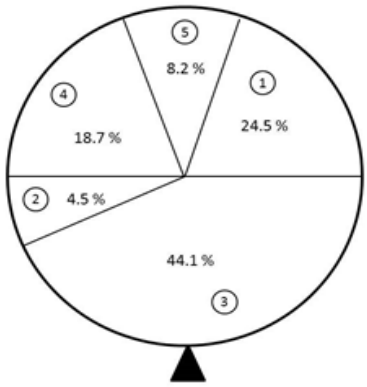

Figura 8.3 Método de selección de la ruleta. Los cinco individuos de la tabla 8.2 se colocan en una rueda, en la cual cada subdivisión corresponde

a su proporción de aptitud. En esta técnica de selección se simula el giro de una ruleta, de tal forma que seleccionamos como padre al individuo donde la ruleta se detiene. Evidentemente, los individuos más aptos tienen mayor probabilidad de ser seleccionados, pero los menos aptos también podrían ser seleccionados como padres, aunque su probabilidad es muy baja.

La porción que cada individuo ocupa en la ruleta corresponde a su aptitud, de tal forma que los individuos más aptos tengan mayor probabilidad de seleccionarse y recombinarse mediante alguno de los varios tipos de cruzas disponibles. Las figuras 8.4 y 8.5 ilustran el funcionamiento de la cruza de uno y dos puntos respectivamente.

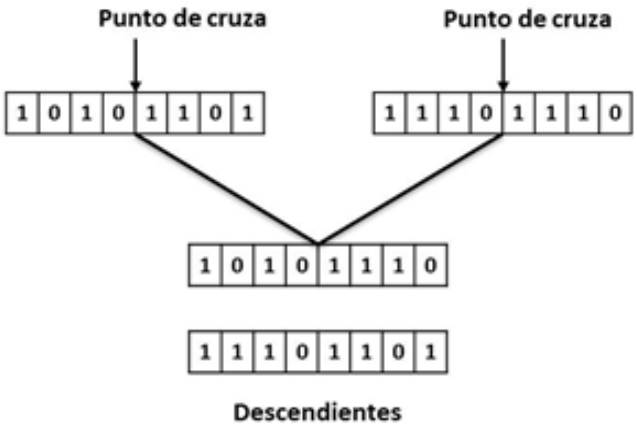

Figura 8.4 Cruza de un punto. Nótese cómo la cruza entre dos padres produce dos hijos. El hijo 1 tiene los primeros 4 bits del padre 1 y los otros 4 del padre 2. El hijo 2 contiene los primeros 4 bits del padre 2 y los otros 4 del padre 1.

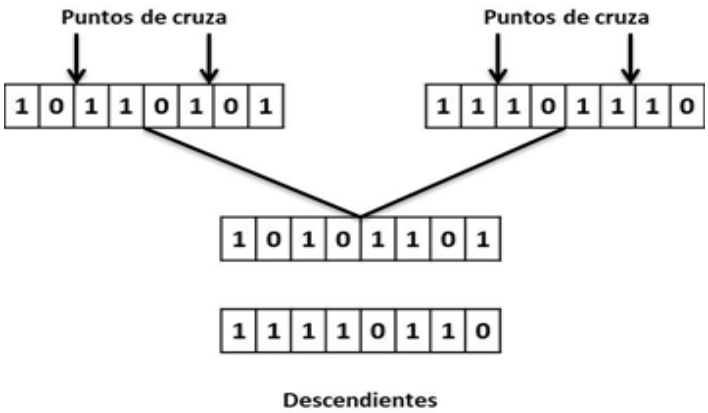

Figura 8.5 Cruza de dos puntos. El hijo tiene los 2 primeros y 2 últimos bits del padre 1 y los bits complementarios del padre 2. El hijo 2 tiene los 2 primeros y 2 últimos bits del padre 2 y los bits complementarios del padre 1.

Finalmente, cada uno de los hijos que se producen mediante la cruza es mutado, como se ilustra en la Figura 8.6.

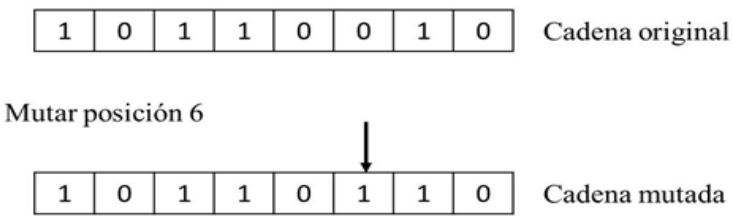

Figura 8.6 Ejemplo del operador de mutación de un algoritmo genético. En este caso, se muta la posición 6 de la cadena original. El operador aplica un "NOT" a la posición original, con lo cual se cambia el cero a uno.

Con los años, se desarrollaron distintas variantes del Algoritmo Genético que involucran el uso de esquemas de selección deterministas y de estado uniforme (en los cuales se reemplaza a un solo individuo de la población en cada iteración, en vez de reemplazarlos a todos), codificaciones no binarias, operadores de cruza y de mutación muy sofisticados y/o altamente especia-

lizados para un dominio en particular, cromosomas diploides y multiploides, así como otro tipo de operadores tales como la inversión, la traslocación y la segregación.[78]

Un aspecto distintivo de los Algoritmos Genéticos es que se ha considerado tradicionalmente que el operador de cruza es su fuente de mayor poder y que el operador de mutación es sólo complementario, como un esquema para mantener diversidad y para mantener el espacio de búsqueda completamente conectado. Esta filosofía se contrapone a la de las Estrategias Evolutivas en las que la mutación es el operador principal y la cruza es sólo un operador secundario y a la Programación Evolutiva en la que sólo se usa la mutación.

Holland en su libro de 1975 trató de modelar el funcionamiento del Algoritmo Genético a través de su "Teorema de los Esquemas" en el que deriva una fórmula que proporciona la probabilidad de que un cierto patrón de bits (denominado "esquema") sobreviva a los efectos de la selección, la cruza y la mutación.[79] Con el paso de los años, se desarrollaron modelos matemáticos mucho más sofisticados para intentar explicar el funcionamiento de un Algoritmo Genético, incluyendo aquellos que se basan en cadenas de Markov y los que se basan en mecánica estadística.[80] La primera demostración de convergencia de un Algoritmo Genético ordinario[81] (llamado "canónico") mostró que debe usarse un operador conocido como "elitismo" para poder garantizar convergencia. Este operador consiste en retener al mejor individuo de cada generación y pasarlo intacto a la generación siguiente, sin cruzarlo ni mutarlo.

---

[78] Goldberg, D. E. **Genetic Algorithms in Search, Optimization, and Machine Learning**, Addison-Wesley, USA, 1989.

[79] Holland, J. H. **Adaptation in Natural and Artificial Systems**, University of Michigan Press, Ann Arbor, Michigan, USA, 1975.

[80] Eiben, A. E., Smith, J. E. **Introduction to Evolutionary Computing**, Natural Computing Series, Springer, Germany, 2003.

[81] Rudolph, G. (1994). **Convergence Analysis of Canonical Genetic Algorithms**. *IEEE Transactions on Neural Networks*, 5:96–101.

En México, se ha hecho investigación muy relevante en torno a los Algoritmos Genéticos, tanto en el contexto de aprendizaje de máquina[82] como en el de optimización mono-objetivo y multi-objetivo.[83] También se hace investigación en torno al uso de algoritmos genéticos y sus variantes en electrónica (el denominado "hardware evolutivo"),[84] visión,[85] bioinformática,[86] finanzas,[87] criptografía,[88] procesamiento de imágenes,[89] y diseño.[90] Asimismo, hay grupos de investigación que han abordado el estudio de los operadores evolutivos[91] y de aspectos teóricos de los algoritmos genéticos.[92] En nuestro país se ha abordado también el estudio de la programación genética,[93] una variante del algoritmo genético propuesta por John Koza[94] en la que se usa una codificación de árbol para evolucionar programas.

8.5. Otras Metaheurísticas Bio-Inspiradas

En México se han estudiado otras metaheurísticas bio-inspiradas entre las que destacan las siguientes:

- **Cúmulos de Partículas:** Propuesta por Kennedy y Eberhart a mediados de los noventa[95] simula el movimiento de un conjunto de par-

[82] Por ejemplo, hay grupos de investigación en el ITESM Campus Monterrey y en el INAOE.

[83] Hay grupos de investigación que han abordado estos temas en la Universidad Veracruzana, la Universidad de Guadalajara, el CINVESTAV-IPN, la Universidad Autónoma de Querétaro, el CICESE, el CIMAT y la UNAM.

[84] En el INAOE y el CIMAT.

[85] En el CICESE.

[86] En el CICESE y el ITESM Campus Estado de México.

[87] En el ITESM Campus Monterrey, el CINVESTAV-IPN y el Banco de México.

[88] En el CIC-IPN y el CINVESTAV-IPN.

[89] En el ITESM Campus Guadalajara, el CIMAT, el CICESE y el CINVESTAV-IPN.

[90] En el Instituto Tecnológico Autónomo de México (ITAM).

[91] En la Universidad de Guadalajara y el CINVESTAV-IPN.

[92] En la UNAM y el CINVESTAV-IPN

[93] En la UNAM, INFOTEC, CINVESTAV-IPN y el CICESE.

[94] Koza, J. R. **Genetic Programming. On the Programming of Computers by Means of Natural Selection**. The MIT Press, Cambridge, Massachusetts, 1992.

[95] Kennedy, J., Eberhart, R.C. **Swarm intelligence**, Morgan Kaufmann, San Francisco, California, USA, 2001.

tículas en un fluído (agua o aire). Las trayectorias se definen a partir de una fórmula muy simple que involucra la velocidad de cada partícula, un factor de inercia y la influencia de la mejor solución que se haya obtenido para una partícula individual y para todo el cúmulo. Hay grupos de investigación que han usado la optimización mediante cúmulos de partículas, sobre todo para optimización mono-objetivo y multi-objetivo.[96]

- **Sistemas Inmunes Artificiales:** Busca simular el comportamiento de nuestro sistema inmune, el cual nos defiende de las enfermedades que invaden nuestro cuerpo en forma de "antígenos". Para ello, el sistema inmune genera células denominadas "anticuerpos", que tienen la capacidad de acoplarse a los antígenos a fin de anularlos. Los primeros modelos computacionales del sistema inmune datan de los ochenta y se popularizaron por Stephanie Forrest, quien realizó aportaciones clave en esta área.[97] Hoy en día existen diversas propuestas de modelos computacionales basados en sistemas inmunes, de entre los que destacan la selección negativa, la selección clonal y el modelo de red inmune. De ellos, la selección clonal ha sido el más utilizado en la literatura especializada (sobre todo para resolver problemas de optimización). Los sistemas inmunes artificiales se han usado por grupos de investigación en México para abordar problemas de criptografía y de optimización mono-objetivo y multi-objetivo.[98]

- **Evolución Diferencial:** Propuesta por Kenneth Price y Rainer Storn a mediados de los noventa,[99] se puede ver más bien como un método de búsqueda directa en el que se trata de estimar el gradiente en una

[96] En CINVESTAV-IPN, unidades Tamaulipas, Guadalajara y Zacatenco.

[97] De Castro, L., Timmis, J. **Artificial Immune Systems: A New Computational Intelligence Approach**, Springer, 2002.

[98] En el ITESM Campus Estado de México, el CIC-IPN y el CINVESTAV-IPN.

[99] Storn, R., Price, K. (1997). **Differential Evolution – a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces**. *Journal of Global Optimization*, 11(4):341–359.

región (y no en un punto) mediante el uso de un operador que recombina a 3 individuos. Pese a que su diseño luce muy simple, esta técnica es sumamente poderosa para realizar optimización en espacios continuos y produce resultados equiparables e incluso superiores a los obtenidos por las mejores Estrategias Evolutivas que se conocen, pero usando menos parámetros y con implementaciones mucho más sencillas. Hay grupos de investigación en México que han utilizado la evolución diferencial en diferentes problemas de optimización mono-objetivo y multi-objetivo.[100]

- **Colonia de Hormigas:** Propuesta por Marco Dorigo a principios de los noventa[101] simula el comportamiento de un grupo de hormigas que sale de su nido a buscar comida. Durante su recorrido segregan una sustancia llamada "feromona" la cual pueden oler las demás hormigas. Al combinarse los rastros de feromona de toda la colonia la ruta resultante es la más corta del nido a la comida. En el modelo computacional desarrollado por Dorigo se considera un factor de evaporación de la feromona y se debe poder evaluar una solución parcial al problema. En su versión original, esta metaheurística se podía utilizar sólo para problemas que se mapearan al problema del viajero. Sin embargo, con los años se desarrollaron versiones más sofisticadas que se pueden aplicar a otros tipos de problemas combinatorios e incluso a problemas continuos. En México, existen grupos de investigación que han utilizado esta metaheurística para resolver problemas de diseño de circuitos y de optimización mono-objetivo y multi-objetivo en espacios continuos.[102]

[100] En la Universidad Veracruzana y el CINVESTAV-IPN.

[101] Dorigo, M., Di Caro, G. (1999). **The ant colony optimization meta-heuristic**. En D. Corne, M. Dorigo y F. Glover (eds), *New Ideas in Optimization*, McGraw-Hill, UK, pp. 11–32.

[102] En CINVESTAV-IPN.

8.6. La Computación Evolutiva en México

Los orígenes de la investigación en computación evolutiva en México se remontan a finales de los años ochenta en el Instituto Tecnológico y de Estudios Superiores de Monterrey (ITESM) Campus Monterrey y en la Universidad Nacional Autónoma de México (UNAM). En el ITESM se creó el Centro de Inteligencia Artificial en 1989 (hoy llamado Centro de Sistemas Inteligentes) el cual, casi desde su inicio, tuvo interés en estudiar Sistemas Inspirados en la Naturaleza. En la UNAM existen tesis de licenciatura y doctorado de finales de los ochenta y principios de los noventa dirigidas por Germinal Cocho Gil, que utilizan el método de Monte Carlo y algoritmos genéticos,[103] si bien las primeras tesis que utilizan el término "algoritmo genético" en su título se remontan a 1993.[104]

Hacia la segunda mitad de los noventa existían ya cursos sobre computación evolutiva en instituciones tales como el Centro de Investigación en Computación del Instituto Politécnico Nacional (CIC-IPN), la Maestría en Inteligencia Artificial de la Universidad Veracruzana y la de la UNAM.

Actualmente, existen diversos posgrados en México en los cuales es posible obtener una maestría y un doctorado en computación con especialidad en Computación Evolutiva. Por ejemplo, en el Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional (CINVESTAV-IPN), el Centro de Investigación en Matemáticas (CIMAT), el Instituto de Investigación en Matemáticas Aplicadas y Sistemas de la UNAM (IIMAS-UNAM),

[103] Miramontes Vidal, O. R. **Algunos Aspectos de la Teoría de Autómatas Celulares y sus Aplicaciones en Biofísica**, Tesis de Licenciatura (Física), Facultad de Ciencias, UNAM, 1988.

Arce Rincón, J. H. **Estados de Mínima Energía en una Red a Temperatura Cero**, Tesis Doctoral (Doctor en Ciencias (Física)), Facultad de Ciencias, UNAM, 1990.

Miramontes Vidal, P. E. **Un Esquema de Autómata Celular como Modelo Matemático de la Evolución de los Ácidos Nucleicos**, Tesis Doctoral (Doctor en Ciencias (Matemáticas)), Facultad de Ciencias, UNAM, 1992.

[104] Comunicación personal de la Dra. Katya Rodríguez Vázquez, quien realizó diversas consultas al sistema TESISUNAM para proporcionar este dato.

el CIC-IPN, la Universidad de Guadalajara, el Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), el Centro de Investigación Científica y de Educación Superior de Ensenada (CICESE), el Centro de Investigación e Innovación en Tecnologías de la Información y Comunicación (INFOTEC) y en el ITESM.

De manera paralela a los trabajos que condujeron a la escritura de este capítulo, la sección de Computación Evolutiva de la Academia Mexicana de Computación creó un directorio (https://www.cs.cinvestav.mx/~compevol/) el cual pretende complementar el contenido de este capítulo con información de las áreas en las que trabajan los investigadores así como sus páginas web, donde puede encontrarse más detalle sobre el trabajo que han realizado y que llevan a cabo actualmente.

8.7. Perspectivas

La investigación en torno a la Computación Evolutiva ha tenido un crecimiento importante en México en los últimos años. Hoy en día existe un número considerable de investigadores que trabajan con distintos tipos de algoritmos evolutivos y con otras metaheurísticas bio-inspiradas, no sólo en torno a aplicaciones, sino también en investigación básica (por ejemplo, diseño de nuevos algoritmos, análisis teórico y estudio de operadores). Como se indicó anteriormente, existen también varios programas de maestría y doctorado reconocidos en el Padrón Nacional de Posgrado de CONACyT en los cuales es posible especializarse en Computación Evolutiva. Asimismo, existen grupos de investigación que colaboran con investigadores de varios países[105] y que tienen amplia visibilidad internacional.

En años recientes ha habido también una creciente participación de estos grupos en los congresos internacionales más importantes del área: la

[105] Todos los grupos de investigación de México mencionados a lo largo de este capítulo han tenido colaboraciones con investigadores de países tales como Estados Unidos, Escocia, Australia, España, Chile, Francia, Holanda, Alemania, China y Japón.

Genetic and Evolutionary Computation Conference, el *IEEE Congress on Evolutionary Computation* y *Parallel Problem Solving from Nature*. Sin embargo, son todavía muy pocos los grupos de investigación de México que publican en las revistas más reconocidas del área: *IEEE Transactions on Evolutionary Computation*, *Evolutionary Computation* y *Genetic Programming and Evolvable Machines*. Esto puede deberse a que varios de los grupos de investigación están conformados por doctores jóvenes en proceso de consolidación y también porque algunos de los grupos actuales están orientados al desarrollo de aplicaciones y prefieren publicar en revistas no especializadas en computación evolutiva.

Finalmente, es deseable que los grupos de investigación en Computación Evolutiva existentes en México que son relativamente recientes se consoliden y que su visibilidad aumente; sin duda hay el potencial para que esto ocurra dada la cantidad de colaboraciones internacionales con las que ya se cuenta.

La Computación en México por especialidades académicas,
se terminó de imprimir en septiembre de 2017 en
Agys Alevín S. C. Retorno de Amores No. 14-102.
Col. Del Valle. C. P. 03100, Ciudad de México.
En su composición se utilizó tipo Garamond.
Impreso en papel couché mate de 115 grs.
La edición consta de 240 ejemplares.