CAPÍTULO 2

TÉCNICAS DE COMPUTACIÓN EVOLUTIVA

"Evolution is cleverer than you are."

FRANCIS CRICK

Elbow Room: the varieties of free will worth wanting,
D. Dennett, 1984

2.1. **Introducción**

Las técnicas de computación evolutiva constituyen un conjunto de heurísticas emergentes, utilizadas exitosamente para la resolución de una variada gama de problemas en las áreas de optimización combinatoria, diseño de artefactos, búsqueda de información, control de dispositivos y aprendizaje automático, entre otros. Estas técnicas basan su operativa en la emulación de los mecanismos de la evolución natural, identificados por Charles Darwin en su célebre obra *El Origen de las Especies por medio de la Selección Natural*: la *selección natural*, la *reproducción* y la *diversidad genética* de individuos (Darwin, 1859).

Las técnicas de computación evolutiva trabajan sobre una *población* compuesta por un conjunto de codificaciones de soluciones candidatas para el problema a resolver. Estas soluciones interactúan entre sí, siguiendo los principios darwinianos de la evolución natural con la idea de producir iterativamente mejores soluciones al problema. Las soluciones potenciales se evalúan mediante una función de adecuación o *función de fitness*, que toma en cuenta el problema que se plantea resolver. En la naturaleza, durante el proceso evolutivo los seres vivos tratan de resolver los problemas relacionados con la supervivencia para garantizar la perpetuación de la especie. Mediante el mecanismo comentado, las técnicas de computación evolutiva emulan el proceso biológico de adaptación de los organismos vivos al entorno y las condiciones del medio, aplicándolo a la resolución de problemas en variadas áreas.

Este capítulo presenta a las técnicas de computación evolutiva en general, y a los algoritmos genéticos en particular, como mecanismos de resolución de problemas de optimización combinatoria, y problemas análogos en otras áreas de aplicación. El capítulo comienza con una introducción a las técnicas de computación evolutiva que presenta sus principales características. A continuación se ofrece un panorama histórico del desarrollo de las técnicas algorítmicas basadas en la emulación de los procesos de la evolución natural, presentando globalmente los modelos más populares de técnicas evolutivas: las *Estrategias de Evolución*, la *Programación Genética* y los *Algoritmos Genéticos*. Las características de los algoritmos genéticos son presentadas en la sección siguiente, explicando los detalles de su funcionamiento y sus ventajas y desventajas respecto a otros métodos exactos y heurísticos de resolución de problemas de optimización.

2.2. Técnicas de Computación Evolutiva

La expresión genérica *computación evolutiva* designa a un amplio conjunto de técnicas heurísticas de resolución de problemas complejos que basan su funcionamiento en un mecanismo análogo a los procesos de la evolución natural. Trabajando sobre un conjunto de soluciones a un problema determinado, la metodología utilizada por estas técnicas se fundamenta en el uso de mecanismos de selección de las mejores soluciones potenciales y de construcción de nuevas soluciones candidatas mediante recombinación de características de las soluciones seleccionadas.

Varios esquemas algorítmicos han sido propuestos para los algoritmos evolutivos. La idea más generalizada sobre el mecanismo de un algoritmo evolutivo que trabaja sobre una población **P** se presenta en la Figura 2.1.

```
Inicializar(P(0))
generación=0;
mientras (no CriterioParada) hacer
    Evaluar(P(generación))
    Padres = Seleccionar(P(generación))
    Hijos = Aplicar Operadores Evolutivos (Padres)
    NuevaPoblacion = Reemplazar(Hijos, P(generación))
    generación ++
    P(generación) = NuevaPoblacion
fin
retornar Mejor Solución Encontrada
```

Figura 2.1: Esquema algorítmico genérico de un algoritmo evolutivo.

El algoritmo evolutivo trabaja sobre individuos que representan potenciales soluciones al problema, codificados de acuerdo a un mecanismo prefijado. Los individuos son evaluados de acuerdo a una *función de fitness* que toma en cuenta la adecuación de cada solución al problema que se intenta resolver.

La operativa del algoritmo evolutivo comienza con una etapa de inicialización de los individuos, que puede ser completamente aleatoria, muestreando al azar diferentes secciones del espacio de soluciones, o guiada de acuerdo a características del problema a resolver. El algoritmo evolutivo podría inclusive tomar como población inicial individuos resultantes como salida de algún otro algoritmo heurístico de resolución que permitiera calcular buenas soluciones iniciales aproximadas para el problema.

La evolución propiamente dicha se lleva a cabo en el ciclo que genera nuevos individuos a partir de la población actual mediante un procedimiento de aplicación de operadores estocásticos. En este ciclo se distinguen cuatro etapas:

- *Evaluación*: etapa que consiste en asignar un valor de adecuación (*fitness*) a cada individuo en la población. Este valor evalúa que tan bien resuelve cada individuo el problema en cuestión, y es utilizado para guiar el mecanismo evolutivo.
- *Selección*: proceso que determina candidatos adecuados, de acuerdo a sus valores de fitness, para la aplicación de los operadores evolutivos con el objetivo de engendrar la siguiente generación de individuos.
- *Aplicación de los operadores evolutivos*: etapa que genera un conjunto de descendientes a partir de los individuos seleccionados en la etapa anterior.
- *Reemplazo*: mecanismo que realiza el recambio generacional, sustituyendo individuos de la generación anterior por descendientes creados en la etapa anterior.

Diversas políticas para la selección y el reemplazo de individuos permiten modificar las características del algoritmo evolutivo. Aplicando políticas adecuadas es posible privilegiar los individuos más adaptados en cada generación (*estrategias de elitismo*), aumentar la presión selectiva sobre individuos mejor adaptados, generar un número reducido de descendientes en cada generación (*modelos de estado estacionario*), y otras muchas variantes.

Los *operadores evolutivos* determinan el modo en que el algoritmo explora el espacio de soluciones del problema. Una gran diversidad de propuestas de operadores evolutivos han surgido en los casi cuarenta años de vida de la computación evolutiva. Los diferentes operadores y las particularidades en su modo de aplicación dan características peculiares a las distintas variantes de algoritmos evolutivos. Los *operadores de recombinación*, que permiten combinar características de dos o más individuos con la idea de obtener descendientes mejor adaptados y los *operadores de mutación*, que introducen diversidad mediante modificaciones aleatorias, son los operadores evolutivos más difundidos.

La condición de parada de la fase iterativa del algoritmo evolutivo usualmente toma en cuenta la cantidad de generaciones procesadas, deteniéndose el ciclo evolutivo al alcanzar un número prefijado de generaciones. Otras alternativas consideran la variación de los valores de fitness –deteniendo el ciclo evolutivo cuando el proceso se estanca y no obtiene mejoras considerables en los valores de fitness– o estimaciones del error cometido respecto al valor óptimo del problema o una aproximación, en caso de conocerse.

2.3. Reseña histórica

Las técnicas de computación evolutiva surgieron sobre 1960, interpretando la naturaleza como una formidable máquina de resolver problemas y tratando de encontrar el origen de dicha potencialidad para utilizarla en la resolución de problemas complejos.

Desde los inicios de la era de la computación varios referentes vislumbraron la idea de aplicar los mecanismos naturales para diseñar dispositivos capaces de evolucionar en su operativa y aplicarse a la resolución de complejos problemas en las áreas de aprendizaje automático, problemas de optimización y búsqueda de información. Luego de la época inicial, en donde se propusieron las primeras ideas sobre el funcionamiento de los mecanismos evolutivos, tres modelos algorítmicos diferenciados se desarrollaron en forma simultánea, aunque en algunas ocasiones han interactuado entre sí. Las *Estrategias de Evolución*, la *Programación Evolutiva* y los *Algoritmos Genéticos* constituyen hoy en día las principales líneas de trabajo en el área de la computación evolutiva. Esta sección presenta un breve resumen histórico de su desarrollo, desde las propuestas pioneras hasta su consolidación como potentes mecanismos de resolución de problemas en los últimos quince años.

2.3.1. Primeras propuestas

Las relaciones detectadas entre los procesos de aprendizaje y la evolución natural dieron el marco para las primeras ideas sobre algoritmos evolutivos en la década de 1950. Ya en 1948 Alan Turing había sugerido la conexión entre ambos aspectos, proponiendo desarrollar programas *automodificables* capaces de jugar ajedrez y simular otras actividades *inteligentes* desarrolladas por los seres humanos, utilizando técnicas evolutivas. Los detalles de las ideas y las propuestas de Turing pueden consultarse en la sección correspondiente en la Enciclopedia de Filosofía de la Universidad de Stanford (Hodges, 2002).

John von Neumann también se interesó en la combinación de técnicas evolutivas y computación, en especial en el caso de los autómatas celulares. Sobre el final de su vida se encontraba trabajando en el área, tal como lo testimonia su texto inconcluso *Teoría de Autómatas AutoReplicables* (Von Neumann, 1966) que sería editado luego de la muerte del autor por su colega A. Burke. Von Neumann propuso mecanismos evolutivos basados en la programación para implementar autómatas con un poder computacional equivalente a una máquina universal de Turing. Además, conjeturó sobre el comportamiento de *poblaciones* de autómatas capaces de abordar problemas complejos *comunicándose* entre sí. Al respecto de las ideas y los trabajos de Von Neumann es posible consultar el texto de Dyson (1999) y la reseña de Mitchell (1998) sobre autómatas celulares.

El matemático y biólogo Nils Barricelli tomó la posta y utilizó la infraestructura montada por Von Neumann en el Instituto para Estudios Avanzados en la Universidad de Princeton, para llevar a cabo la primera simulación de "vida artificial" basada en principios evolutivos, en el período comprendido entre los años 1953 y 1956. Su trabajo consistió en diseñar un modelo computacional para la evolución darwiniana e investigar el rol de la simbiogénesis en el origen y desarrollo de la vida. La simbiogénesis considera a los organismos vivientes como una sucesión de asociaciones simbióticas entre formas sencillas de vida. Al respecto de su trabajo, Barricelli comentó: "La distinción entre la evolución entre números en una computadora y entre nucleótidos en un laboratorio químico es muy sutil" (citado por Dyson (1999)). Barricelli trabajó directamente sobre código de máquina para implementar un "universo" compuesto de 512 celdas a ser ocupadas por números de 8 bits a los cuales denominó *genes*. Un conjunto de reglas evolutivas gobernaban la propagación de los números en la grilla..

Con su simulación de vida artificial, Barricelli sentó el precedente sobre el uso de los métodos evolutivos para la resolución de problemas. En su trabajo valoró adecuadamente la utilidad de los mecanismos de recombinación para mejorar la adaptación de individuos, indicando que el proceso evolutivo basado exclusivamente en la selección por aptitud y un operador probabilístico de mutación de individuos no era capaz en general de producir buenos resultados en tiempos razonables, y que un operador de recombinación, que según Barricelli debía estar basado en conceptos de simbiogénesis, era necesario para acelerar la evolución.

Un estupendo y ameno resumen del trabajo de Barricelli se presenta en el texto de Dyson (1999). Los conceptos de la simbiogénesis y su relación con la computación evolutiva pueden consultarse en los trabajos recientes de Watson y Pollack (1999, 2000) y Corning (1998).

Uno de los primeros intentos de aplicar técnicas evolutivas para la resolución de problemas prácticos de ingeniería, se propuso en el área de control estadísticos de procesos. En 1957, George Box propuso modificar los sistemas de operaciones estáticas tradicionales por mecanismos dinámicos capaces de realizar ajustes en las variables de control, evaluar sus efectos y modificar el proceso para mejorar los resultados obtenidos, siguiendo una analogía con el desarrollo de los procesos químicos en la naturaleza. Aplicó su idea a procesos de manufactura, proponiendo el método Evolutionary Operation –EVOP–, que introduce variaciones deliberadas en los parámetros tratando de mejorar el objetivo del proceso. Para evitar cambios abruptos en la calidad de los productos, se limita a introducir pequeños cambios en las variables de control.

Debido a limitaciones al poder computacional, el esquema de Box nunca fue implementado en la práctica como un algoritmo para computadora, pero en ciertos casos la técnica fue aplicada con controles de calidad realizados por un *comité técnico de evaluación* compuesto por personas idóneas. Con el posterior incremento en la habilidad de los procesadores para asimilar, manejar y evaluar grandes volúmenes de información, la técnica recibió interés creciente. Al respecto del método Evolutionary Operation pueden consultarse los propios trabajos de Box (1957, 1998). Asimismo, el texto de Goldberg (1989a) brinda una breve descripción de la técnica propuesta por Box, aplicada al control de un sencillo proceso dependiente de tres variables.

Otras técnicas que simulaban a la evolución natural para intentar el aprendizaje automático de dispositivos, simular entornos de vida biológica o resolver problemas genéricos fueron propuestas a fines de la década de 1950.

R. Friedberg trabajó en 1958 con un sistema de recompensas para calificar instrucciones que influían en la calidad de los resultados de programas sencillos (Friedberg, 1958). El mecanismo de aprendizaje se basaba, al igual que en la propuesta de Box, en realizar pequeñas modificaciones aleatorias y luego evaluar los programas modificados. Friedberg llegó a evolucionar secuencias de instrucciones –codificadas en lenguaje de máquina– que realizaban cómputos modestos, capaces de realizar operaciones como el desplazamiento de un bit desde una celda de memoria a otra. El enfoque de Friedberg contaba con la particularidad de representar comportamientos mediante un procedimiento generalizado (es decir, a través de programas completos) y de proponer modificaciones basadas en procedimientos muy especializados. Esta característica ha sido indicada como un defecto por otros investigadores en el área de la Inteligencia Artificial (Mc Carhty, 1990), dado que pequeñas modificaciones en la operativa de una máquina no pueden relacionarse en general con pequeñas modificaciones a su programa. Otros investigadores formularon críticas más duras al trabajo de Friedberg: M. Minsky argumentó que un método basado en una búsqueda puramente aleatoria era mucho mejor que el algoritmo de Friedberg, mientras que J. McCarthy y H. Simon propusieron seguir un enfoque más relacionado con los procesos de la evolución natural, duplicando subrutinas, realizando copias modificadas y dejando otras sin modificar.

Aunque Friedberg nunca consideró estrictamente que su trabajo fuera una emulación de la evolución natural, es posible interpretar de este modo su propuesta. El sistema de recompensas de Friedberg se basaba en la idea de aplicar un mecanismo de selección de instrucciones asociado con la frecuencia con la cual producían resultados exitosos. Para mayores detalles sobre las ideas de Friedberg es posible consultar los artículos de Mc Carthy (1990, 2003).

Una *simulación de sistemas genéticos* utilizando computadoras fue realizada por S. Fraser en 1957. Aunque su línea de trabajo era la biológica, Fraser logró interpretar parte de la potencialidad de los sistemas computacionales para emular sistemas biológicos evolutivos. Pero analizando el mecanismo evolutivo desde su punto de vista biológico, no fue capaz de proponer una extensión de su algoritmo a condiciones artificiales o resolución de problemas no relacionados con la biología (Fraser, 1957).

En la década de 1960, A. Newell y H. Simon propusieron un sistema al que llamaron “solucionador general de problemas” (*General Problem Solver*) que permitía al usuario especificar un escenario compuesto por *objetos* y definir *operadores* a aplicar sobre los objetos. El sistema se mostró capaz de resolver problemas sencillos definidos en espacios con número reducido de parámetros, utilizando técnicas heurísticas especificadas por el programador y que permitían al método evolucionar sus resultados utilizando un método al estilo del ensayo y error. Algunos argumentos formulados contra el mecanismo evolutivo del *General Problem Solver* se centraron el criticar el hecho de que la "inteligencia" del sistema estuviera completamente determinada por las heurísticas especificadas por el programador y que el sistema no era capaz de tomar "iniciativas" por su propia cuenta, al no incluir operadores evolutivos genéricos. De todos modos, el *General Problem Solver* consistió en una de las primeras propuestas de un sistema capaz de hallar soluciones genéricas, independientes del dominio de los problemas a resolver (Newell y Simon, 1963).

Los estudios de W. Bledsoe y H. Bremermann son generalmente mencionados como los trabajos precursores del concepto moderno de Algoritmos Genéticos. Estos reconocidos científicos, reconocidos como los "padres de la inteligencia artificial" junto con J. McCarthy, M. Minsky y S. Amarel entre otros, fueron capaces de interpretar la potencialidad de los mecanismos evolutivos para la resolución de problemas. Respecto a su trabajo con Bledsoe, Bremermann ofrece una descripción de los primeros intentos de aplicar técnicas evolutivas para mejorar algoritmos de reconocimiento de patrones que utilizaban redes neuronales artificiales: "Sobre 1960-61 experimentamos con la idea de aplicar *algoritmos genéticos* para optimizar la eficiencia de los procesos de percepción: tratar los pesos de las conexiones sinápticas como nucleótidos de ADN; mutarlos, recombinarlos y seleccionarlos, como en la evolución darwiniana ... El método funcionó en principio, pero el problema de entrenamiento de las redes neuronales resultó tener extraordinarios requerimientos computacionales, ya que involucraba el uso de un gran número de conexiones sinápticas con pesos asociados" (Bremermann y Anderson, 1991).

De acuerdo a Goldberg (1989a) y De Jong et al. (1997), Bledsoe y Bremermann sugirieron la idea de codificación binaria, y el uso de un valor de aptitud en la aplicación de los algoritmos evolutivos para la resolución de problemas de optimización numérica. Bledsoe propuso el esquema de generar individuos, aplicar mutaciones y seleccionar los que produjeran mejores resultados y Bremermann lo extendió para considerar poblaciones de individuos. (Boyer et al, 1996, 1998). Bremermann notó la importancia del operador de mutación para evitar el estancamiento del proceso de búsqueda en mínimos locales del problema. El primer resultado teórico sobre la operativa de los algoritmos evolutivos fue alcanzado por Bremermann al determinar la probabilidad de mutación óptima para resolver problemas linealmente separables, presentado en Bremermann et al. (1965).

Aunque la mayoría de sus conceptos se utilizan en la actualidad, y de hecho Bremermann es aceptado como el "creador" de los algoritmos genéticos (Fogel y Anderson, 2000), en su trabajo obtuvo algunos resultados decepcionantes al aplicarlos para optimizar funciones lineales y convexas. Con frecuencia sus algoritmos evolutivos no lograban converger a la solución del problema, y requerían ser complementados con otras heurísticas para lograr buenos resultados. En la actualidad se conoce que precisamente en esos dominios de trabajo, los algoritmos evolutivos no son capaces de competir con las técnicas heurísticas tradicionales de optimización, pero en contrapartida son capaces de obtener buenos resultados para numerosos tipos de problemas donde las heurísticas fallan con frecuencia.

2.3.2. Estrategias de Evolución

Las estrategias de evolución –*Evolutionsstrategie*– fueron introducidas por Rechenberg en 1965. En su propuesta inicial, consistía en un método de optimización que trabajaba sobre individuos compuestos por números reales para optimizar parámetros en problemas de diseño en ingeniería (Rechenberg, 1965). El método modelaba la evolución al nivel de los propios valores a optimizar –sin utilizar codificaciones específicas– aplicando un mecanismo de selección determinística y un operador de mutación aleatoria basada en distribuciones de probabilidad, en general distribuciones gaussianas.

A partir de esa primera propuesta, los métodos de estrategias de evolución se han difundido ampliamente. La utilización del operador de mutación como base del mecanismo evolutivo caracteriza a esta familia de métodos, aunque en versiones recientes se han propuesto modelos que incorporan mecanismos de recombinación como operador secundario, como la familia de *cruzamientos aritméticos* (Back et al, 1997).

La propuesta inicial de Rechenberg fue desarrollada luego por Schwefel en la década de 1970. Su modelo de evolución característico, guiado por el operador de mutación, los distingue de las otras técnicas evolutivas, y por este motivo fueron estudiados de forma independiente. En su versión más simple, el proceso evolutivo se basa en la generación de un descendiente por parte de un individuo padre, mediante el operador de mutación, reemplazando el nuevo individuo generado a su progenitor en la población. Dos modelos avanzados de Estrategias de Evolución fueron formulados por Schwefel en su tesis doctoral de 1975. Ambas variantes trabajan con un conjunto de padres que genera un conjunto de descendientes y se diferencian por el modo de reemplazar los individuos progenitores por sus descendientes. Estos modelos avanzados son actualmente conocidos como Estrategias de Evolución (μ,λ) y Estrategias de Evolución $(\mu+\lambda)$. En ambos modelos μ padres generan λ descendientes, pero mientras en el modelo (μ,λ) la selección se realiza solamente entre los descendientes, en el modelo $(\mu+\lambda)$ los padres y los descendientes compiten entre sí (Schwefel, 1975).

Ninguno de los modelos de estrategias de evolución fue estudiado con profundidad ni aplicados hasta fines de los años 80, cuando la influencia del poder de cómputo de las computadoras paralelas hizo resurgir el interés por esta técnica evolutiva. Diversos esquemas han sido propuestos en los años 90, aplicando conceptos de paralelismo, optimización multiobjetivo y esquemas de adaptación. Sobre 1995 fueron presentados nuevos resultados teóricos sobre los modelos $(\mu+\lambda)$ y (μ,λ) (Back et al, 1997).

2.3.3. Programación Evolutiva

Las técnicas de programación evolutiva fueron propuestas por Fogel, Owens y Walsh en 1966 como un método para evolucionar comportamientos de autómatas de estado finito utilizados para la predicción de series temporales (Fogel et al, 1966). El enfoque presentado en el artículo referido ha sido considerado "moderno", en el sentido de que sigue la línea de trabajo actual de la computación evolutiva. Introduce el concepto de población y dimensiona adecuadamente la importancia del mecanismo de selección en el proceso evolutivo, tomando en cuenta el "entorno" al cual se encuentran expuestos los individuos. El método fue aplicado con éxito para la resolución de problemas sencillos, aunque no faltó la crítica habitual sobre que la técnica propuesta no era superior que una búsqueda completamente aleatoria.

En 1975, Holland sugirió que un algoritmo evolutivo podría ser utilizado con una codificación que representara un programa (Holland, 1975). Luego de una década en el silencio, la programación evolutiva se renovó a mediados de 1980, modificando su planteo para permitir representaciones que extendieron su aplicabilidad para la resolución de variados problemas de optimización. En la década de 1980, algunos investigadores retomaron la sugerencia de Holland y realizaron implementaciones de programas evolutivos aplicados a heurísticas de optimización (Smith, 1985), al dilema del prisionero (Fujicki, 1986), y a la evolución de expresiones LISP para programas capaces de resolver juegos sencillos (Hicklin, 1986). Estos trabajos y otras aplicaciones de la programación evolutiva se presentan en el artículo de P. Angeline (1998), que ofrece una perspectiva histórica de las propuestas de estructuras ejecutables evolutivas.

La principal línea de trabajo de programas evolutivos se basa en el trabajo de John Koza, quien le dio el nombre de *programación genética*. Koza propuso un mecanismo evolutivo para la generación de programas, extendiendo las ideas previas de Cramer respecto al uso de árboles de parsing para representar programas (Koza, 1996). A partir de la propuesta de Koza, diversos investigadores han estudiado sobre temas relacionados con la programación genética, incluyendo la aplicabilidad de la técnica a un amplio conjunto de problemas, fundamentos teóricos del mecanismo, diseño de operadores evolutivos adecuados y aplicación de técnicas de procesamiento paralelo, entre otros (O' Reilly, 1995).

En los últimos años, las líneas de trabajo en el área de la programación evolutiva se han concentrado principalmente en aplicaciones como el entrenamiento de redes neuronales y la evolución de sistemas difusos. Algunos nuevos fundamentos matemáticos han sido presentados por parte del creador de las técnicas (Fogel, 1995).

2.3.4. Algoritmos Genéticos

Aún reconociendo los trabajos pioneros de Bremermann, en la práctica se considera a Holland como el fundador de los Algoritmos Genéticos, por ser quien formalizó su mecanismo de funcionamiento en su texto *Adaptation in Natural and Artificial Systems* (Holland, 1975). El trabajo de Holland se remonta a la década de 1960, donde comenzó su estudio de las características de los sistemas adaptativos. A lo largo de sus publicaciones es posible reconocer que sus modelos iban tomando las características de los modernos algoritmos genéticos, al trabajar con representaciones (genomas) y operadores de mutación, cruzamiento e inversión.

Otros investigadores trabajaron con Holland en la Universidad de Michigan realizando sus tesis de doctorado en el área de los nóveles algoritmos genéticos. Bagley propuso utilizarlos para determinar los parámetros en funciones de evaluación de programas capaces de jugar juegos sencillos. Rosenberg y Weinberg los aplicaron en la simulación de organismos *alife* o de vida artificial. Cavicchio trabajó sobre problemas de reconocimiento de patrones, siguiendo las ideas de Bledsoe y Browning. Hollstein aplicó algoritmos genéticos a un problema de optimización matemática pura. Frantz estudió el fenómeno de la epístasis, el efecto posicional y los operadores de inversión, y colaboró proponiendo nuevos operadores evolutivos. Todos estos trabajos son comentados en detalle en el *Handbook of Evolutionary Computation* (Back et al, 1997), y en el texto de Goldberg (1989a).

La tesis doctoral de De Jong (1975) presentó una serie de experimentos computacionales aplicados a la optimización de un conjunto de funciones que, con el transcurrir del tiempo, se consolidaron como el conjunto de funciones de prueba estándar para determinar la utilidad de las técnicas evolutivas. De Jong combinó sus análisis con los resultados teóricos presentados por Holland sobre el mecanismo de búsqueda de los algoritmos genéticos –el *teorema de los esquemas*– definiendo las bases teóricas para el estudio de las técnicas evolutivas.

En la década de 1980 los algoritmos genéticos fueron ganando popularidad como estrategia de resolución de problemas combinatorios y de diseño. El interés creciente y la formación de una comunidad de investigadores en el área condujeron a la realización de la primera Conferencia Internacional en Algoritmos Genéticos en 1985. La publicación del texto de Goldberg (1989a) contribuyó a difundir aún más esta técnica evolutiva, presentando los fundamentos teóricos y una variada gama de aplicaciones en las áreas de optimización, búsqueda y aprendizaje. Desde entonces, los intereses de la comunidad científica que trabaja en el área de algoritmos genéticos se han diversificado, extendiendo los análisis teóricos, proponiendo nuevos modelos y abordando una amplia gama de aplicaciones.

La próxima sección presenta en detalle las características de los algoritmos genéticos y las particularidades de su mecanismo evolutivo.

2.4. Algoritmos Genéticos

Los algoritmos genéticos constituyen una de las técnicas de computación evolutiva más difundidas en la actualidad, como consecuencia de su versatilidad para resolver un amplio rango de problemas. Al constituir un caso de técnica evolutiva, los algoritmos genéticos basan su operativa en una emulación de la evolución natural de los seres vivos, trabajando sobre una población de soluciones potenciales evoluciona de acuerdo a interacciones y transformaciones únicas. Los individuos que constituyen la población se esfuerzan por sobrevivir: una selección programada en el proceso evolutivo, inclinada hacia los individuos más aptos, determina aquellos individuos que formarán parte de la siguiente generación. El grado de adaptación de un individuo se evalúa de acuerdo al problema a resolver, mediante la definición de una función de adecuación al problema, la *función de fitness*. Bajo ciertas condiciones, el mecanismo definido por los operadores inspirados por la genética natural y la evolución darwiniana lleva a la población a converger hacia una solución aproximada al óptimo del problema, luego de un determinado número de generaciones.

La formulación tradicional de un algoritmo genético fue presentada de excelente manera en el texto de Goldberg (1989a), el cual popularizó el uso de los Algoritmos Genéticos para una variada gama de problemas en las áreas de búsqueda, optimización y aprendizaje automático.

En su formulación clásica, los algoritmos genéticos se basan en el esquema genérico de un Algoritmo Evolutivo presentado en la Figura 2.1. A partir de este esquema, el algoritmo genético define "Operadores Evolutivos" que implementan la recombinación de individuos (el operador de *cruzamiento*) y la variación aleatoria para proporcionar diversidad (el operador de *mutación*), resultando el esquema presentado en la Figura 2.2.

```
Inicializar(P(0))
generación=0
mientras (no CriterioParada) hacer
    Evaluar(P(generación))
    Padres = Seleccionar(P(generación))
    Hijos = Aplicar Recombinación (Padres)
    Hijos = Aplicar Mutación (Hijos)
    NuevaPob = Reemplazar(Hijos, P(generación))
    generación ++
    P(generación) = NuevaPob
fin
retornar Mejor Solución Encontrada
```

Figura 2.2: Esquema algorítmico de un algoritmo genético.

La característica distintiva de los algoritmos genéticos respecto a las otras técnicas evolutivas consiste en su uso fundamental del cruzamiento como operador principal, mientras que la mutación se utiliza como operador secundario tan solo para agregar una nueva fuente de diversidad en el mecanismo de exploración del espacio de soluciones del problema. Inclusive la mutación puede llegar a ser un operador opcional o estar ausente en algunas variantes de algoritmos genéticos que utilizan otros operadores para introducir diversidad.

En general, los algoritmos genéticos se han utilizado para trabajar con codificaciones binarias para problemas de búsqueda en espacios de cardinalidad numerable, aunque su alto nivel de aplicabilidad ha llevado a proponer su trabajo con codificaciones reales, e inclusive con codificaciones no tradicionales, dependientes de los problemas a resolver.

Algunos autores consideran que el uso del mecanismo de selección denominado *selección proporcional*, que determina la cantidad de copias de individuos a considerar en la recombinación de forma proporcional a sus valores de fitness, es una característica que distingue a los algoritmos genéticos (Back et al, 1997). Pero tomando en cuenta la diversidad de mecanismos de selección utilizados en las propuestas de algoritmos genéticos en los últimos años es posible concluir que si bien en general se recurre a la selección proporcional, otros mecanismos son igualmente utilizados para modificar el proceso de exploración del espacio de soluciones de los diferentes problemas abordados.

2.4.1. Representación de soluciones

Los algoritmos genéticos no trabajan directamente sobre las soluciones del problema en cuestión, sino que lo hacen sobre una abstracción de los objetos solución, usualmente denominadas *cromosomas* por analogía con la evolución natural biológica. Un cromosoma es un vector de *genes*, mientras que el valor asignado a un gen se denomina *alelo*.

En la terminología biológica, *genotipo* denota al conjunto de cromosomas que definen las características de un individuo. El genotipo sometido al medio ambiente se denomina *fenotipo*. En términos de los algoritmos genéticos el genotipo también esta constituido por cromosomas, utilizándose generalmente un único cromosoma por individuo solución al problema. Por ello suelen utilizarse indistintamente los términos *genotipo*, *cromosoma* e *individuo*. Por su parte, el *fenotipo* representa un punto del espacio de soluciones del problema.

Dado que un algoritmo genético trabaja sobre *cromosomas*, se debe definir una función de *codificación* sobre los puntos del espacio de soluciones, que mapea todo punto del espacio de soluciones en un genotipo, tal como se indica en la Figura 2.3. La función inversa de la codificación, denominada *decodificación* permite obtener el fenotipo asociado a un cromosoma.

$$\textit{Codificación} : \textit{Espacio de soluciones} \rightarrow \textit{cromosoma}$$

Figura 2.3: Especificación de la función de codificación de un algoritmo genético.

Tomando en cuenta la observación anterior, los mecanismos de codificación de individuos solución resultan importantes para el proceso de búsqueda de los algoritmos genéticos. Habitualmente los algoritmos genéticos utilizan codificaciones *binarias* de largo fijo. Los individuos se codifican por un conjunto de cardinalidad conocida de valores binarios (ceros y unos) conocido como *string de bits* o *bitstring*. Cada *bitstring* representa a una solución potencial del problema de acuerdo al mecanismo de codificación predefinido, en general dependiente del problema.

Otros esquemas de codificación han sido utilizados con menor frecuencia en los algoritmos genéticos. En particular las codificaciones basadas en números reales son útiles para representar soluciones cuando se resuelven problemas sobre espacios de cardinalidad no numerable, como en el caso de determinación de parámetros en problemas de control o entrenamiento de redes neuronales. Los esquemas basados en permutaciones de enteros son útiles para problemas de optimización combinatoria que involucran hallar ordenamientos óptimos, como los problemas de *scheduling* o el reconocido Traveling Salesman Problem. Codificaciones dependientes de los problemas se han propuesto con frecuencia, como un mecanismo que permite incorporar conocimiento específico en la resolución de problemas complejos.

La Figura 2.4 resume gráficamente la relación entre el espacio de soluciones de un problema, la población de cromosomas con la cual trabaja el algoritmo genético y las funciones de codificación y decodificación.

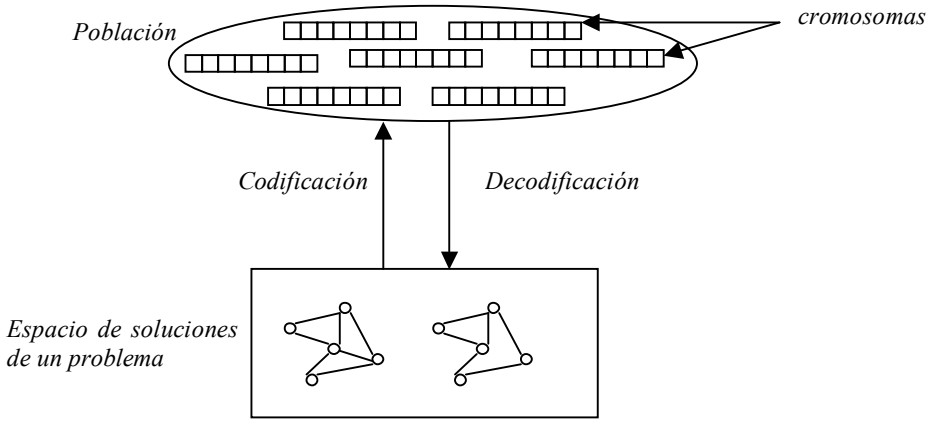

Figura 2.4: Codificación de soluciones en un algoritmo genético.

2.4.2. Función de fitness

Todo cromosoma tiene un valor asociado de *fitness* que evalúa aptitud del individuo para resolver el problema en cuestión. La función de fitness tiene el mismo tipo que la función objetivo del problema, lo cual implica que el cálculo del valor de fitness se realiza sobre el fenotipo correspondiente al cromosoma, como se presenta en la Figura 2.5.

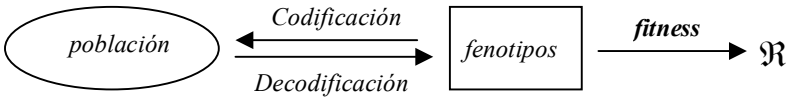

Figura 2.5: Codificación y función de fitness.

La función de fitness tiene una influencia importante en el mecanismo del AG. Si bien actúa como una caja negra para el proceso evolutivo, la función de fitness guía el mecanismo de exploración, al actuar representando al entorno que evalúa la bondad de un individuo solución para la resolución del problema.

Varios puntos han sido identificados como importantes al momento de construir la función de fitness. De acuerdo a Alba y Cotta (2003), es posible indicar que:

- La función de fitness debe contemplar el criterio del problema de optimización (minimización o maximización de un objetivo) y las restricciones presentes en el problema de optimización. En caso de surgir soluciones no factibles del problema, la función de fitness deberá asignarle valores adecuados que garanticen que tales individuos no se perpetúen durante el proceso evolutivo (valores muy pequeños en el caso de un problema de maximización y muy elevados en el caso de un problema de minimización). A tales efectos diversas transformaciones a aplicar sobre funciones de fitness han sido definidas para mapear problemas de minimización a maximización y aplicar penalizaciones a soluciones no factibles (Goldberg, 1989).

- Deben contemplarse los casos en que el entorno presente problemas para la evaluación de la función de fitness, utilizando evaluaciones parciales cuando existen valores de fitness no definidos. Asimismo, debe considerarse el uso de funciones de fitness multivaluadas que asignen diferentes valores a un mismo individuo para simplificar la labor del operador de selección.
- El caso de funciones de fitness que varían dinámicamente durante la evolución del algoritmo genético debe tenerse en cuenta. En este caso mecanismos complejos como las representaciones múltiples son útiles para introducir memoria en la operativa del algoritmo genético.
- En general, la función de fitness será compleja de evaluar, en particular demandará un esfuerzo computacional mucho mayor que el requerido para realizar los operadores evolutivos. Inclusive podría ocurrir que el proceso de evaluación sea tan complejo que solamente valores aproximados pudieran obtenerse en tiempos razonables. Este aspecto será importante al momento de proponer técnicas de alta performance para mejorar la eficiencia de los algoritmos genéticos.
- En caso de problemas con objetivos múltiples, todos ellos deben estar contemplados en la función de fitness. La utilización de algoritmos genéticos para la resolución de problemas multiobjetivo constituye en sí misma una subárea de investigación con complejidades inherentes.
- Para resolver problemas de dominancia de soluciones muy adaptadas en generaciones tempranas de la evolución, y para evitar el estancamiento en poblaciones similares al final de la evolución, deben considerarse mecanismos de *escalado* de los valores de fitness (Michalewicz, 1992).

2.4.3. Operadores

La gran mayoría de las variantes de algoritmos genéticos utiliza como principales operadores a la selección, la recombinación y la mutación.

El mecanismo de selección determina el modo de perpetuar *buenas* características, que se asumen son aquellas presentes en los individuos más adaptados. El mecanismo de *selección proporcional* o *selección por ruleta* elige aleatoriamente individuos utilizando una ruleta sesgada, en la cual la probabilidad de ser seleccionado es proporcional al fitness de cada individuo. Otros mecanismos de selección introducen diferentes grados de elitismo, conservando un cierto número prefijado de los mejores individuos a través de las generaciones. En el caso de la *selección por torneo* se escogen aleatoriamente un determinado número de individuos de la población, los cuales compiten entre ellos para determinar cuáles se seleccionarán para reproducirse, de acuerdo a sus valores de fitness. El mecanismo de *selección basado en el rango* introduce el mayor grado de elitismo posible, al mantener entre generaciones un porcentaje generalmente elevado, de los mejores individuos de la población. Estas diferentes políticas de selección, conjuntamente con políticas similares utilizadas para determinar los individuos reemplazados por los descendientes generados posibilitan el diseño de diferentes modelos evolutivos para los algoritmos genéticos.

Los esquemas de codificación binaria de largo fijo tienen como ventaja principal que resulta sencillo definir operadores evolutivos simples sobre ellos. En la formulación clásica de un algoritmo genético, denominada por Goldberg (1989b) como *Algoritmo Genético Simple*, se propone como operador de recombinación el *cruzamiento de un punto*, que consiste en obtener dos descendientes a partir de dos individuos padres seleccionando un punto al azar, cortando los padres e intercambiando los trozos de cromosoma, tal como se presenta en la Figura 2.6.

Figura 2.6: Esquema del cruzamiento de un punto.

El operador tradicional de mutación introduce diversidad en el mecanismo evolutivo, simplemente modificando aleatoriamente uno de los valores binarios del cromosoma. Sobre un esquema de codificación binaria la modificación consiste en invertir el valor binario de un alelo, y por ello recibe el nombre de mutación de inversión del valor de un bit (*flip–bit*); su operativa se ejemplifica en la Figura 2.7.

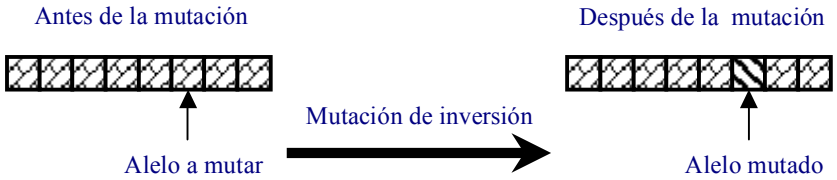

Figura 2.7: Esquema de la mutación de inversión del valor de un alelo.

Tanto la recombinación como la mutación son operadores probabilísticos, en el sentido en que se aplican o no, teniendo en cuenta una tasa de aplicación del operador. Generalmente la tasa de aplicación del operador de cruzamiento es elevada en un algoritmo genético simple (entre 0,5 y 0,9) mientras que la tasa de aplicación del operador de mutación es muy baja, del orden de 0,001 para cada bit en la representación.

Operadores evolutivos más complejos han sido propuestos como alternativas para modificar el comportamiento del mecanismo de exploración del espacio de soluciones. Es habitual encontrar operadores de cruzamiento *multipunto* en donde se utilizan dos o más puntos de corte (la operativa del cruzamiento de dos puntos se ejemplifica en la Figura 2.8) o *uniformes* en donde para cada posición en el cromosoma se decide intercambiar material genético de acuerdo a una probabilidad prefijada (su operativa se ejemplifica en la Figura 2.9).

Figura 2.8: Esquema del cruzamiento de dos puntos.

Figura 2.9: Esquema del cruzamiento uniforme.

2.4.4. Modelos de evolución

En el modelo tradicional de algoritmos genéticos el paso básico de evolución consiste en el reemplazo de la totalidad de la población por sus descendientes en cada generación (*modelo generacional*). Otros modelos diferentes de evolución han sido propuestos por los investigadores en el área. Entre ellos, es posible destacar el modelo *de estado estacionario* (Whitley, 1991). En este paradigma se crea un único nuevo individuo en cada generación, tratando de balancear el compromiso entre exploración (dada por el conjunto de individuos que forman la población) y explotación (realizada por el mecanismo de generación del único individuo creado en cada generación, generalmente utilizando técnicas de elitismo) para la resolución del problema. Un modelo intermedio entre el tradicional y el de estado estacionario es el *modelo de gap generacional* (De Jong, 1981), en el cual un porcentaje de la población (denominado *gap*) se genera en cada paso evolutivo.

Otros modelos de evolución más complejos han sido formulados para los algoritmos genéticos (modelos jerárquicos, no heterogéneos, híbridos, inspirados en los métodos de las estrategias de evolución). Estos modelos se diferencian de los tradicionales, incluyendo complejidades inherentes al mecanismo evolutivo que proponen. En este trabajo presentaremos una de las variantes no tradicionales de algoritmos genéticos, el algoritmo CHC (*Cross generational elitist selection, Heterogeneous recombination and Cataclysmic mutation*) propuesto por Eshelman (1991).

2.4.5. Formalización del mecanismo de los algoritmos genéticos

La teoría tradicional de exploración del espacio de soluciones por parte del mecanismo evolutivo de los algoritmos genéticos fue formalizada por Holland en 1975. Holland definió el concepto de *esquema*, para representar a conjuntos de individuos con características comunes de codificación y enunció el *teorema de los esquemas* para formalizar el muestreo de hiperplanos del espacio de soluciones. Como consecuencia de este teorema se propuso la denominada *hipótesis de los building blocks* que fundamenta el éxito del mecanismo de exploración de los algoritmos genéticos en el número exponencial de muestras que reciben aquellos esquemas cortos, con pocas posiciones definidas, pero que tienen un valor promedio de fitness superior al del resto de esquemas de la población. El mecanismo de recombinación, fundamental en el proceso evolutivo de un algoritmo genético, tenderá a crear individuos cada vez más aptos, perpetuando aquellas características adecuadas y combinándolas con otras de similar calidad (Holland, 1975). Trabajos teóricos posteriores han complementado las estimaciones originales de Holland (Whitley, 1997), e inclusive se han presentado formalizaciones matemáticas exactas del mecanismo (Poloni, 1999, 2000), aunque también ofrecido puntos de vista alternativos y críticos a la *hipótesis de los building blocks* (Thornton, 1998).

Goldberg (1991) demostró que el tiempo que un algoritmo genético emplea en converger hacia una solución única depende del tamaño de la población considerada. Utilizando los mecanismos de selección adecuados –aquellos que favorecen la reproducción de los mejores individuos–, el tiempo de convergencia del algoritmo genético es del orden de O($n \log n$), siendo n el tamaño de la población utilizada.

Adicionalmente a sus estudios sobre la formación de bloques básicos de construcción de soluciones, Holland se percató que el mecanismo evolutivo de los algoritmos genéticos tiene una característica denominada *paralelismo intrínseco*, mediante la cual el ciclo evolutivo procesa un número mayor de esquemas que la cantidad de individuos presentes en la población. En efecto, trabajando sobre una población de n individuos, y por tanto requiriendo un esfuerzo computacional para realizar solamente n evaluaciones de la función de fitness, el algoritmo genético procesa útilmente O(n^3) esquemas en cada generación. Por este motivo, al examinar un elevado número de secciones del espacio de soluciones, el mecanismo de los algoritmos genéticos logra compensar la disrupción de esquemas de mayor largo y de alto número de posiciones definidas cuando se aplican los operadores evolutivos de cruzamiento y mutación.

2.4.6. Comparación de los algoritmos genéticos con otras técnicas heurísticas

Si bien no existe una opinión unánime sobre la aplicabilidad de los algoritmos genéticos, la amplia gama de problemas resueltos exitosamente en diversas áreas han popularizado a esta técnica evolutiva como una de las más versátiles y robustas. Las opiniones de los investigadores en el área concuerdan en señalar a los algoritmos genéticos como altamente útiles para resolver problemas con espacio de soluciones de dimensión elevada o no muy bien comprendido de antemano, donde el diseño de operadores específicos de *hill-climbing* no sea sencillo, en problemas multimodales donde los métodos heurísticos tradicionales pueden quedar atrapados en óptimos locales o en casos donde no es necesario obtener una solución óptima al problema, sino que una buena solución aproximada sería suficiente.

Los algoritmos genéticos presentan ciertas características que los hacen ventajosos cuando se los compara con otras técnicas de resolución de problemas. La operativa del mecanismo evolutivo de los algoritmos genéticos es independiente de particularidades del dominio, permitiendo definir esquemas genéricos capaces de abordar diferentes clases de problemas. Esta característica, conjuntamente con el hecho de que el mecanismo evolutivo se aplique sobre representaciones, hace a los algoritmos genéticos aplicables a una amplia gama de problemas.

Además, los algoritmos genéticos no son sensibles a efectos de no linearidad de la función a optimizar o características del problema que afectan a algoritmos estándar de optimización que utilizan información adicional, como las técnicas de hill-climbing o algoritmos que asumen aspectos de linearidad, convexidad, diferenciabilidad u otras características sobre la función a optimizar. Estas característica brindan a los algoritmos genéticos una significativa robustez de funcionamiento y de aplicabilidad.

Desde el punto de vista de la resolución del problema, el algoritmo genético funciona como una caja negra que solamente utiliza como dato de entrada la función de fitness definida para el problema para evaluar a los individuos en el ciclo evolutivo. No utiliza información adicional sobre las características del problema, aunque ciertas particularidades de la codificación pueden complementar la operativa simple de los operadores evolutivos y dirigir la búsqueda. La independencia del mecanismo de búsqueda evolutivo de las características del problema permite a los algoritmos genéticos evitar, en ocasiones, el estancamiento en mínimos locales del problema en los cuales son proclives a caer ciertos algoritmos tradicionales basados en gradientes u otras búsquedas locales dirigidas.

Una descripción detallada del mecanismo operativo de los algoritmos genéticos, que complemente los breves detalles que se resumen en este capítulo, puede encontrarse en los textos de Goldberg (1989a) o de Mitchell (1996).

2.5. Conclusiones

En los últimos quince años, las técnicas de computación evolutiva se han consolidado como un conjunto de metaheurísticas efectivas y robustas como consecuencia de su alta aplicabilidad a la resolución de problemas en múltiples áreas.

Tomando en cuenta los numerosos y exitosos antecedentes que refieren a la utilización de técnicas evolutivas para la resolución de problemas de optimización combinatoria, surgen como una alternativa natural para la resolución de los complejos problemas de optimización subyacentes a los problemas de diseño de redes de comunicaciones confiables, que constituyen el objetivo de esta Tesis.

Este capítulo ha presentado los conceptos genéricos vinculados con las técnicas de computación evolutiva y ha comentado en detalle las características de los algoritmos genéticos en particular, al considerarlos como mecanismos de resolución de problemas de optimización combinatoria y problemas análogos en otras áreas de aplicación.

2.6. Nota sobre las referencias bibliográficas

Las referencias bibliográficas mencionadas en este capítulo son muy diversas y varias de ellas han sido tomadas de citas de los textos de Goldberg (1989a), Mitchell (1996) y Dyson (1999). Otros documentos son citados y comentados en las reseñas de Angeline (1998) y Mitchell (1998) y en el *Handbook of Evolutionary Computation* (Back et al, 1997). Se presentan a continuación los detalles de las referencias, indicándose las citas correspondientes en aquellos casos de artículos y textos no consultados directamente.