Capítulo IV

ALGORITMOS GENÉTICOS

4.1. INTRODUCCIÓN

La estructura de un algoritmo genético simple se corresponde con la estructura de cualquier programa evolutivo (ver figura 1.1). Durante la iteración *t*, un algoritmo genético mantiene una población de soluciones potenciales (cromosomas, vectores), $P(t) = \{x^t_1, ..., x^t_n\}$. Cada solución x^t_n se evalúa para dar alguna medida de su fitness. Entonces, se forma una nueva población (iteración $t + 1$) al seleccionar los individuos más adaptados. Algunos miembros de esta nueva población se someten a alteraciones por medio de crossover y mutación, para formar nuevas soluciones. El crossover combina las características de dos cromosomas padres para formar dos hijos similares al intercambiar segmentos entre los padres. La idea detrás de la aplicación del operador de crossover es el intercambio de información entre diferentes soluciones potenciales. La mutación altera arbitrariamente uno o más genes del cromosoma seleccionado, estos cambios aleatorios se realizan con probabilidad usualmente baja. La idea detrás del operador de mutación es introducir alguna variabilidad en la población.

Para resolver un problema particular, un algoritmo genético (como cualquier programa evolutivo) deberá tener los siguientes componentes:

- ✓ Una representación genética de las soluciones potenciales del problema.
- ✓ Una forma de crear la población inicial de soluciones potenciales.
- ✓ Una función de evaluación que juegue el papel de medio ambiente, y permita ordenar las soluciones de acuerdo a su fitness.
- ✓ Operadores genéticos que alteren la composición de los hijos.
- ✓ Valores de varios parámetros que el algoritmo genético utiliza.

Grefenstette y Baker [81] presentan una versión modificada de la descripción de un algoritmo genético. Esta descripción difiere del paradigma presentado por Holland, donde la selección se hace para obtener padres para recombinación [84].

procedure: Algoritmo Genético
begin
$t \leftarrow 0$;
iniciar *P(t);*
evaluar *P(t);*
while no(condición de terminación) **do**
recombinar *P(t)* para producir *C(t)*;
evaluar *C(t)*;
seleccionar *P(t+1)* de *P(t)* y *C(t)*;
$t \leftarrow t + 1$;
end while
end procedure

Figura 4.1 Estructura general de un algoritmo genético.

Sea $P(t)$ y $C(t)$ la población de padres e hijos en la generación actual t, la estructura general de un algoritmo genético se describe en la figura 4.1.

La iniciación generalmente se realiza en forma aleatoria. La recombinación típicamente involucra crossover y mutación para generar los hijos. De hecho, sólo hay dos clases de operaciones en algoritmos genéticos:

- ✓ *Operaciones genéticas*: crossover y mutación.
- ✓ *Operación de evolución*: selección.

Las operaciones genéticas imitan el proceso de herencia de genes para crear nuevos hijos en cada generación. La operación de evolución imita el proceso de evolución de Darwin para crear poblaciones de generación en generación.

4.1.1. VOCABULARIO GA

Como los GAs tienen sus raíces tanto en la genética natural como en la ciencia de la computación, la terminología usada en la literatura de algoritmos genéticos es una mezcla de la natural y de la artificial.

En un organismo biológico, la estructura que codifica la prescripción especificando como está construido el organismo se llama *cromosoma*. Se pueden necesitar uno o más

Algoritmos Genéticos	Explicación
Cromosoma (string, individuo)	Solución
Genes (bits)	Parte de una solución
Locus	Posición del gen
Alelos	Valores del gen
Fenotipo	Solución decodificada
Genotipo	Solución codificada

Tabla 4.1 Explicación de los términos de algoritmos genéticos

cromosomas para especificar el organismo completo. El conjunto de cromosomas se llama *genotipo*, y el organismo resultante, *fenotipo*. Cada cromosoma comprende un conjunto de estructuras individuales llamadas *genes*. Cada gen codifica una característica particular de un organismo, y la ubicación del gen, *locus*, dentro de la estructura del cromosoma determina la característica particular que representa el gen. En un locus particular, un gen puede codificar cualquiera de los distintos valores de la característica que representa. Los diferentes valores de un gen se denominan *alelos*.

La tabla 4.1 [69] muestra la correspondencia entre los términos de algoritmos genéticos y los de optimización.

4.1.2. REPRESENTACIÓN

Hay un conjunto de mecanismos de representación (también llamada codificación) que se usan para resolver distintos problemas de optimización en GAs, entre ellas se encuentran la codificación binaria, real, permutaciones, etc.

4.1.2.1. CODIFICACIÓN BINARIA

La codificación binaria es una elección excelente para problemas en los cuales un individuo se mapea naturalmente en un string de ceros y unos. Un string binario se usa como un cromosoma para representar valores reales de la variable x. El largo del vector depende de la precisión requerida. Por ejemplo E = [10001010] es un string binario con ocho genes. El locus del i-ésimo gen es simplemente el i-ésima posición en el string y su valor o alelo lo da $E[i]$.

Para muchos problemas una codificación binaria no es apropiada debido a [24]:

- ✓ *Epístasis*: significa una fuerte interacción entre genes en un cromosoma. En otras palabras, la epistasis mide el grado en el cual la contribución en el fitness de un gen depende de los valores de otros genes.
- ✓ *Representación natural*: el problema a ser resuelto requiere un conjunto de símbolos de orden mayor.
- ✓ *Soluciones ilegales*: los operadores genéticos pueden producir soluciones ilegales con una codificación binaria, ya que una codificación binaria puede no describir naturalmente un punto del espacio de búsqueda.

4.1.2.2. CODIFICACIÓN REAL

En problemas de optimización de funciones reales se da una función *n*-dimensional, por ejemplo $f(x, y, z)$. El objetivo de optimización es típicamente el requerimiento para ubicar el valor máximo (o mínimo) de la función en un dominio dado. En GAs clásicos, usados en problemas de optimización de funciones reales, una solución potencial al problema se codifica en un string de bits. En un ejemplo tridimensional $f(x, y, z)$, una solución potencial podrá ser $f(x_1, y_1, z_1)$ y x_1, y_1, z_1 se codificarán como substrings tridimensionales. Esos substrings se concatenarán para formar un único genotipo de strings de bits. Cada parámetro se puede codificar como un único string binario o como un número en punto flotante tradicional.

El principal objetivo detrás de esta implementación es que el algoritmo genético esté más cerca del espacio del problema. Esto fuerza, pero también permite, que los operadores sean más específicos del problema. Por ejemplo, esta representación tiene la propiedad que dos puntos cercanos en el espacio de representación pueden también ser cercanos en el espacio del problema, y viceversa. Esto no es generalmente verdad en la opción binaria, donde la distancia en una representación es normalmente definida por el número de posiciones de bits diferentes. Sin embargo, es posible reducir tal discrepancia usando *codificación Gray*.

4.1.2.3. CODIFICACIÓN CON PERMUTACIONES

Los problemas de permutaciones necesitan el arreglo óptimo de un conjunto de símbolos en una lista. El TSP es uno de esos problemas donde se puede usar un símbolo para identificar una ciudad, y la disposición de los símbolos en una lista representa el orden en el cual el vendedor visita cada ciudad para formar un circuito con todas las ciudades. Una codificación con permutaciones se puede representar por medio de una lista de valores enteros distintos, por ejemplo x = [4, 3, 0, 1, 2] [82, 80, 144, 119, 118, 120]. Cada valor entero en la lista codifica directamente el orden relativo de algún objeto específico del problema. Esta representación prohibe valores de alelos duplicados o perdidos; permite el uso de operadores genéticos de alta performance (tal como el *edge recombination operator* [144]); y facilita un mecanismo de decodificación simple desde el genotipo al fenotipo para el TPS.

4.1.2.4. OTRAS REPRESENTACIONES

Se han usado otras estrategias de codificación en algoritmos evolutivos. Entre ellas se puede incluir representación con árboles [90], matrices [22, 15, 140], y codificaciones con estructuras heterogéneas [71].

4.1.3. CROSSOVER

El crossover es el principal operador genético. Es un operador sexual. Trabaja sobre dos cromosomas a la vez y genera hijos al recombinar ambas características de los cromosomas. Una forma simple de realizar el crossover deberá consistir en elegir en forma aleatoria un punto de corte, y generar el hijo combinando el segmento de un padre a la izquierda del punto de corte con el segmento a la derecha del otro padre. Este método trabaja con la representación de strings de bits y se ha extendido a otras representaciones. La performance del algoritmo genético depende, en gran parte, del comportamiento del operador de crossover usado.

La *probabilidad de crossover* (indicada por p_c) se define como la relación entre el número de hijos producidos en cada generación y el tamaño de la población (generalmente indicado por *pop_size*). Esta probabilidad controla el número esperado de cromosomas que se someten a la operación de crossover. Una alta probabilidad de crossover permite una mayor exploración del espacio de soluciones, reduciendo las

chances de establecerse en un óptimo falso; pero si la probabilidad es muy alta, provoca un gran desperdicio en cuanto a cantidad de tiempo de computación en la exploración de regiones no prometedoras del espacio de soluciones. Los valores propuestos para la probabilidad de crossover son $p_c = 0.6$ [29], $p_c = 0.95$ [78], y $p_c \in [0.75,0.95]$ [124]. El crossover tradicional de un punto introducido por Holland [84] elige un posición de crossover $i \in \{1, ..., l\text{-}1\}$ dentro del string de bits (de longitud l) en forma aleatoria e intercambia los bits a la derecha de esa posición entre ambos individuos. Si los padres se representan por los siguientes vectores $(s_1, s_2,\ldots, s_n)$ y $(v_1, v_2,\ldots, v_n)$, entonces al cruzar los cromosomas luego del punto i se deberán producir los siguientes hijos:

$$(s_1, s_2,\ldots, s_{i-1}, s_i, v_{i+1}, \ldots, v_n) \text{ y } (v_1, v_2,\ldots, v_{i-1}, v_i, s_{i+1}, \ldots, s_n)$$

Una generalización a crossover *multi punto* [43] permite más de un punto de crossovers. En la figura 4.2 se esquematiza el efecto del crossover de 5 puntos.

Al incrementar el número de puntos de crossover, se obtiene el operador de *crossover uniforme* [134]. En el caso de crossover uniforme, el intercambio de segmentos se reduce a bits; por cada bit se decide en forma aleatoria si se realiza el intercambio o no. Comparando este operador con el crossover de un punto, Syswerda reporta mejores resultados con crossover uniforme sobre un conjunto de funciones de prueba.

4.1.4. MUTACIÓN

La mutación es un operador de background en los GAs el cual produce cambios aleatorios en varios cromosomas. Una forma simple de realizar la mutación deberá ser alterar uno o más genes. En un algoritmo genético, la mutación cumple el rol de reposición de genes perdidos en la población durante el proceso de selección y de provisión de aquellos genes que no están presentes en la población inicial.

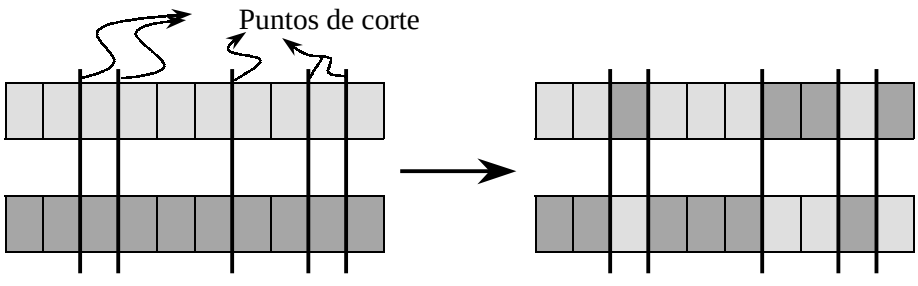

Figura 4.2 Aplicación del crossover de 5 puntos.

La *probabilidad de mutación* (indicada por p_m) se define como el número total de genes en la población que deben ser mutados. La probabilidad de mutación controla el porcentaje en el cual se introducen nuevos genes en la población. Si es muy baja, muchos genes que podrían haber sido producidos nunca se prueban. Si es muy alta, habrá mucha perturbación aleatoria, los hijos comenzarán a perder su parecido a los padres. El algoritmo perderá la habilidad de aprender de la historia de la búsqueda. Valores comunes para la probabilidad de crossover son $p_m = 0.001$ [29], $p_m = 0.01$ [78] y $p_m \in [0.005,0.01]$ [124].

Mientras la mutación en un algoritmo genético sirve como un operador para reintroducir alelos perdidos en la población, es decir posiciones de bits que convergen a un cierto valor en la población completa y por consiguiente no podrían ser recuperados nuevamente por medio de la recombinación, el operador de crossover es el operador de búsqueda más importante de un GA. La idea de crossover es que los segmentos significativos de diferentes padres se combinen para producir un nuevo individuo que se beneficie con combinaciones de bits ventajosas de ambos padres. De esta forma, se espera el surgimiento de segmentos grandes de alto fitness, finalmente trayendo a una solución general buena [84].

4.1.5. EXPLORACIÓN Y EXPLOTACIÓN

La búsqueda es una de los métodos más universales de resolución de problemas, donde no se puede determinar a priori la secuencia de pasos que lleva a la solución. La búsqueda se puede realizar con *estrategias ciegas* o con *estrategias heurísticas* [16]. Las estrategias de búsqueda ciegas no usan información respecto del dominio del problema. Las estrategias de búsqueda heurísticas usan información adicional para guiar la búsqueda. Hay dos cuestiones importantes en estrategias de búsqueda: explotar la mejor solución y explorar el espacio de búsqueda [18]. Michalewicz da una comparación de la búsqueda hill-climbing, búsqueda aleatoria y búsqueda genética [102]. Hill-climbing es un ejemplo de una estrategia que explota la mejor solución por una posible mejora, mientras ignora la exploración del espacio de búsqueda. La búsqueda aleatoria es un ejemplo de una estrategia que explora el espacio de búsqueda mientras ignora la explotación de las regiones prometedoras del espacio de búsqueda. Los algoritmos genéticos son una clase de métodos de búsqueda de propósito general, el

cual combina elementos de búsqueda estocástica y dirigida, las cuales pueden hacer un balance entre exploración y explotación del espacio de búsqueda. Al comienzo de la búsqueda genética existe una población diversa y aleatoria, el operador de crossover tiende a realizar una búsqueda general al explorar todo el espacio de soluciones. Mientras se desarrollan soluciones con fitness alto, el operador de crossover provee exploración en el vecindario de cada una de ellas. En otras palabras, la clase de búsqueda (exploración o explotación) que un operador de crossover realice deberá estar determinada por el ambiente del sistema genético (la diversidad de la población), pero no por el operador en sí mismo. Además, los operadores genéticos simples se diseñan como métodos de búsqueda de propósito general (métodos de búsqueda independientes del dominio), esencialmente realizan una búsqueda a ciegas y podrían no garantizar la producción de mejores hijos.

4.1.6. BÚSQUEDA BASADA EN LA POBLACIÓN

Generalmente, el algoritmo para resolver problemas de optimización es una secuencia de pasos los cuales convergen asintóticamente a la solución óptima. La mayoría de los métodos de optimización clásicos, generan una secuencia determinística de operaciones basadas sobre el gradiente o derivadas de ordenes superiores de la función objetivo. Los métodos se aplican a un único punto del espacio de búsqueda. El punto se mejora gradualmente con direcciones ascendentes y descendentes a través de las iteraciones. Este método punto a punto tiene la desventaja de caer en óptimos locales. Los GAs realizan una búsqueda en múltiples direcciones al mantener una población de soluciones potenciales. La opción población a población procura que la búsqueda escape de óptimos locales. La población se somete a una evolución simulada: en cada generación las soluciones relativamente buenas se reproducen, mientras las soluciones relativamente malas mueren. Los algoritmos genéticos usan reglas de transición probabilísticas para seleccionar quien se reproducirá y quien morirá.

4.1.7. META-HEURÍSTICAS

Al principio los GAs se crearon como una herramienta genérica para la resolución de muchos problemas difíciles. En la mayoría de los primeros trabajos en GAs se usó la

representación interna universal consistente de strings binarios de longitud fija, con operadores genéticos binarios para trabajar de una manera independiente del dominio sin tener conocimiento de la interpretación fenotípica del string. Esta universalidad se reflejó en un fuerte énfasis sobre el diseño de sistemas robustos adaptables para una gran gama de aplicaciones. Sin embargo, los GAs simples son difíciles de aplicar directa y exitosamente en muchos problemas de optimización de resolución complicada. Se han creado varias implementaciones no estándares para problemas particulares en las cuales los GAs se usan como *meta-heurísticas*.

4.2. CODIFICACIÓN DEL PROBLEMA

Cómo codificar en un cromosoma una solución del problema es una cuestión clave para un algoritmo genético. En el trabajo de Holland, la codificación se realiza por medio de strings binarios. Para muchas aplicaciones de GAs, especialmente para problemas de ingeniería industrial, el GA simple tiene una aplicación difícil en forma directa porque el string binario no es una codificación natural. Se han desarrollado muchas técnicas de codificación no string para problemas particulares, por ejemplo *codificación de números reales* para problemas de optimización con restricciones y *codificación con enteros* para problemas de optimización combinatoria. La elección de una representación apropiada de las soluciones candidatas al problema a manejar es fundamental para aplicar GAs, a fin de resolver problemas del mundo real. La representación condiciona todos los pasos subsecuentes del algoritmo genético. En muchos casos de aplicación, es necesario realizar un análisis cuidadoso para asegurar una representación apropiada de soluciones junto a operadores genéticos específicos y útiles al problema.

Una de las características básicas de los algoritmos genéticos es que trabajan sobre un *espacio codificado* y un *espacio de soluciones* en forma alternativa: los operadores genéticos trabajan sobre el espacio codificado, genotipos (cromosomas), mientras la evaluación y la selección lo hacen sobre el espacio de soluciones (fenotipos) (figura 4.3 [69]). La selección natural es la conexión entre cromosomas y la performance de sus soluciones decodificadas. Para la opción no string, surgen tres puntos vinculados con el codificado y decodificado entre cromosomas y soluciones (o la traslación entre fenotipo y genotipo):

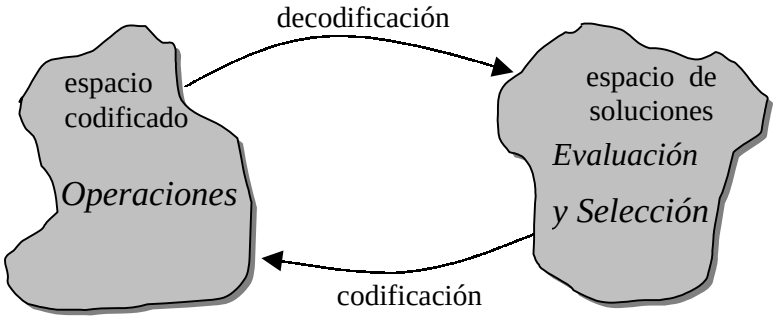

Figura 4.3 Espacio codificado y espacio de soluciones.

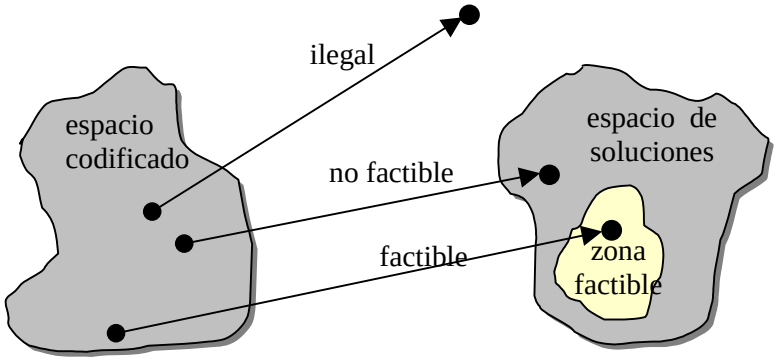

Figura 4.4 Factibilidad y legalidad.

- ✓ La factibilidad de un cromosoma.
- ✓ La legalidad de un cromosoma.
- ✓ La unicidad de la traslación.

La *factibilidad* hace referencia a si una solución decodificada cae en una región factible del problema dado. La *legalidad* hace mención al fenómeno de si un cromosoma representa una solución al problema dado (figura 4.4 [69]).

La no factibilidad de un cromosoma se origina por la naturaleza de los problemas de optimización con restricciones. Todos los métodos, los convencionales o los algoritmos genéticos, deberán manejar restricciones. Para muchos problemas de optimización, la región factible se puede representar por medio de un sistema de igualdades o desigualdades (lineales o no lineales). Para tales casos, se han propuesto muchos métodos de penalidades eficientes para tratar con cromosomas no factibles [67, 103, 130]. En problemas de optimización con restricciones, el óptimo típicamente se produce

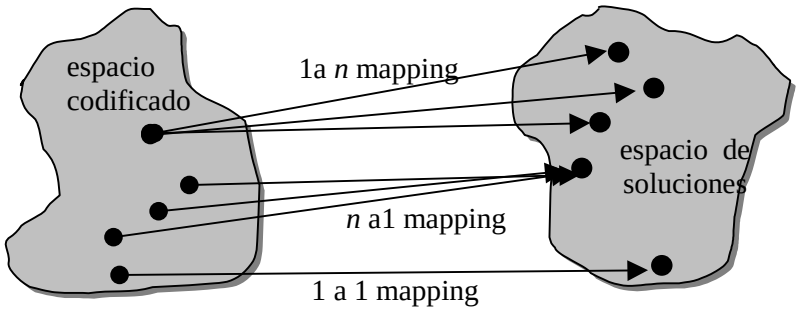

Figura 4.5 La transformación de cromosomas a soluciones.

en el límite entre áreas factibles y no factibles. Las opciones de penalidad forzarán la búsqueda genética para aproximarse al óptimo desde las regiones factibles y no factibles.

La ilegalidad de los cromosomas se origina por la naturaleza de las técnicas de codificación. En muchos problemas de optimización combinatoria, se usan codificaciones específicas del problema y generalmente producen un hijo ilegal por la simple operación de crossover de un punto. Como un cromosoma no se puede decodificar a una solución, esto significa que tampoco se puede evaluar; por lo tanto la opción con penalidades no es aplicable en esta situación. Usualmente, se adoptan técnicas reparadoras para convertir un cromosoma ilegal en uno legal. Por ejemplo, el operador PMX es esencialmente una clase de crossover de dos puntos para representaciones con permutaciones, junto con un procedimiento de reparación para resolver la ilegitimidad causada por el crossover de dos puntos. Orvosh y Davis [112] han mostrado que, para muchos problemas de optimización combinatoria, es relativamente fácil reparar un cromosoma ilegal o no factible; la estrategia de reparación supera a otras estrategias como las de rechazo o las de penalización.

La transformación de un cromosoma en una solución (decodificación) puede pertenecer a alguno de los siguientes tres casos (figura 4.5 [69]):

1. Maping 1 a 1.
2. Maping *n* a 1.
3. Maping 1 a *n*.

El mapping 1 a 1 es el mejor de los tres casos y el mapping 1 a *n* es el menos deseado.

4.3. CONVERGENCIA PREMATURA

La teoría de GAs provee algunas explicaciones de porqué, para una formulación del problema, se puede obtener una convergencia a un punto óptimo buscado. Desgraciadamente, las aplicaciones prácticas no siempre siguen la teoría, las razones principales son:

- ✓ La codificación del problema frecuentemente mueve al GA a trabajar en un espacio diferente al del problema en sí mismo.
- ✓ Hay un límite sobre el número de iteraciones hipotéticamente no limitadas.
- ✓ Hay un límite sobre el tamaño de la población hipotéticamente no limitada.

Una de las implicaciones de esas observaciones es la inhabilidad de los algoritmos genéticos, bajo ciertas condiciones, para hallar las soluciones óptimas; tal característica es la causa para una convergencia prematura a un óptimo local. La convergencia prematura es un problema común de los algoritmos genéticos y de otros algoritmos de optimización. Si la convergencia ocurre muy rápidamente, entonces frecuentemente se pierde el desarrollo de información valiosa en parte de la población. Las implementaciones de algoritmos genéticos son propensas a la convergencia prematura antes de hallar la solución óptima.

Algunos investigadores relacionan la convergencia prematura con dos puntos muy relacionadas:

- ✓ La magnitud y clase de errores introducidos por los mecanismos de muestreos.
- ✓ Las características de la función.

Hay dos cuestiones muy importantes en el proceso de evolución de la búsqueda genética: la *diversidad poblacional* y la *presión selectiva*. Esos factores están fuertemente relacionados: un incremento en la presión selectiva decrementa la diversidad de la población, y viceversa. En otras palabras, una fuerte presión selectiva origina la convergencia prematura de la búsqueda; una débil presión selectiva puede hacer la búsqueda inefectiva [102]. Es importante un balance entre esos dos factores; los mecanismos de muestreos intentan realizar este objetivo. En [144] se observa que esos factores son primarios en la búsqueda genética y en algún sentido, es otra variación de la idea de *exploración* versus *explotación* discutida por Holland y otros. Muchos de los parámetros que se usan para ajustar la búsqueda son realmente medios indirectos para afectar la presión selectiva y la diversidad en la población. Cuando se incrementa la

presión selectiva, la búsqueda se enfoca en los individuos más adaptados de la población, pero por la explotación se pierde diversidad genética. Reduciendo la presión selectiva (usando grandes poblaciones) se incrementa la exploración porque hay más genotipos involucrados en la búsqueda.

4.4. SELECCIÓN

El principio detrás del algoritmo genético es esencialmente la selección natural de Darwin. La selección provee la fuerza motora a un algoritmo genético. La presión selectiva es crítica. En un extremo, la búsqueda terminará prematuramente; mientras que en el otro, el progreso será más lento que el necesario. Típicamente, se sugiere una baja presión selectiva al comienzo de los algoritmos genéticos en favor de una amplia exploración del espacio de búsqueda, mientras se recomienda una alta presión selectiva al final, para explotar las regiones más prometedoras del espacio de búsqueda. La selección dirige la búsqueda del GA a través de regiones prometedoras del espacio de búsqueda.

Se presentan tres cuestiones básicas relacionadas con la fase de selección [68]:

- ✓ *Espacio de muestreo.*
- ✓ *Mecanismo de muestreo*
- ✓ *Probabilidad de selección.*

Cada una de ellas ejerce una influencia significativa sobre la *presión selectiva* y por consiguiente en el comportamiento del algoritmo genético.

4.4.1. ESPACIO DE MUESTREO

El procedimiento de selección puede crear una nueva población para la próxima generación basándose en todos los padres e hijos o parte de ellos. Esto da lugar al problema de espacio de muestreo. Un espacio de muestreo se caracteriza por dos factores: *tamaño* e *integrantes* (padres o hijos). Sea *pop_size* que indica el tamaño de la población y *off_size*, la cantidad de hijos producidos en cada generación. El espacio de muestreo regular tiene el tamaño de *pop_size* y contiene todos los hijos pero sólo una parte de los padres. El espacio de muestreo extendido tiene el tamaño de *pop_size* + *off_size* y contiene el conjunto total de padres e hijos.

4.4.1.1.ESPACIO DE MUESTREO REGULAR

En el algoritmo genético original de Holland, los padres se reemplazan por sus hijos tan pronto estos últimos nacen. Esto se denomina *reemplazo generacional*. Como las operaciones genéticas son ciegas por naturaleza, los hijos pueden ser peores que sus padres. Con la estrategia de reemplazar cada padre por su hijo directamente, algunos cromosomas con fitness alto se pueden perder en el proceso de evolución. Para solucionar este problema, se propusieron varias *estrategias de reemplazo*. Holland sugiere que cada vez que nace un hijo, éste reemplace un cromosoma de la población elegido en forma aleatoria [84]. De Jong propone una estrategia de *crowding* [29]. En el modelo de crowding, cuando nace un hijo, se selecciona un padre para morir, y es aquel que más se parezca al nuevo hijo. En el trabajo de Holland, la selección hace referencia a la elección de padres para recombinación; se forma una nueva población al reemplazar los padres con sus hijos. Esto recibe el nombre de plan reproductivo. Desde el trabajo de Grefenstette y Baker [81], la selección se usa para formar la próxima generación, usualmente con un mecanismo probabilístico. Michalewicz [102] da una descripción detallada de algoritmos genéticos simples donde los hijos reemplazan a sus padres en cada generación; la próxima generación se forma utilizando *selección por ruleta* (*roulette wheel selection*) [102].

4.4.1.2.ESPACIO DE MUESTREO EXTENDIDO

Cuando la selección trabaja sobre un espacio de muestreo extendido, tanto los padres como los hijos tienen las mismas chances de competir por la supervivencia. El caso típico es la selección $(\mu + \lambda)$ [57]. Esta estrategia fue inicialmente usada en estrategias evolutivas [126, 127]. Bäck y Hoffmeister introdujeron este tipo de selección en su algoritmo genético [5, 9]. Con esta estrategia, μ padres y λ hijos compiten por la supervivencia y se seleccionan los μ mejores individuos entre los hijos y los padres como padres de la próxima generación. Otro caso desde las estrategias evolutivas es la selección (μ,λ), la cual selecciona los μ mejores hijos como padres de la próxima generación $(\mu < \lambda)$. Ambos métodos son completamente determinísticos y se pueden convertir en métodos probabilísticos. Aunque la mayoría de los métodos de selección se basan en el espacio de muestreo regular, es fácil implementarlos sobre espacios de

muestreos extendidos. Una ventaja que presenta esta opción es que se puede mejorar la performance del GA al incrementar las probabilidades de crossover y mutación.

4.4.2. MECANISMOS DE MUESTREO

Los mecanismos de muestreos están relacionados con el problema de cómo seleccionar cromosomas del espacio de muestreo. Se han usado tres opciones básicas para sacar muestras de cromosomas [69]:

- ✓ *Muestreos estocásticos.*
- ✓ *Muestreos determinísticos.*
- ✓ *Muestreos mixtos.*

4.4.2.1. MUESTREOS ESTOCÁSTICOS

Una característica común de los métodos encasillados en esta categoría es que la fase de selección determina el número de copias que cada cromosoma recibirá basándose en su probabilidad de supervivencia [11]. De este modo la fase de selección está compuesta de dos partes:

- ✓ Determinar el valor esperado de hijos del cromosoma.
- ✓ Convertir el valor esperado en un número de hijos.

Un valor esperado de hijos del cromosoma es un número real que indica el número promedio de hijos que un cromosoma deberá recibir. El procedimiento de muestreo se usa para convertir el valor esperado real en número de hijos.

El método más conocido en esta clasificación es la *selección proporcional* de Holland o *roulette wheel selection*. La idea es determinar la *probabilidad de selección* (también llamada probabilidad *de supervivencia*). Para el cromosoma *k* con fitness f_k, su probabilidad de selección p_k se calcula de la siguiente manera:

$$p_k = \frac{f_k}{\sum_{j=1}^{pop\_size} f_j}$$

Entonces se puede construir una ruleta en función a esas probabilidades. El proceso de selección está basado en hacer girar la rueda de ruleta *pop_size* veces; cada vez, se selecciona un único cromosoma para la nueva población.

Este mecanismo falla en el caso de fitness negativo o tareas de minimización, si *f* representa los valores de la función objetivo sin realizar un escalamiento apropiado. Por otra parte, cualquier técnica de escalamiento manipula las probabilidades de selección y por consiguiente el mecanismo de selección, de tal forma que los fundamentos teóricos de la selección proporcional se vuelven cuestionables en estos casos.

Baker propone *stochastic universal sampling* [11], el cual usa una única pasada por la rueda (*wheel spin*). La rueda se construye como una ruleta y se gira con un conjunto de marcadores separados a igual distancia, la cantidad de marcadores es igual al tamaño de la población. El valor esperado e_k para el cromosoma k se calcula como $e_k = pop\_size \times p_k$. El procedimiento de la figura 4.6 describe el stochastic universal sampling.

La consideración básica de esta opción es tomar el número esperado de copias de cada cromosoma en la próxima generación. Un punto interesante es la *prohibición de cromosomas duplicados* en la población. Hay dos razones para usar esta estrategia:

- ✓ Prevenir que *super cromosomas* dominen la población al mantener muchas copias en la población. Esto es una causa rápida de convergencia a un óptimo local.

```
procedure: stochastic universal sampling
begin
   sum ← 0;
   ptr ← rand(); // rand() retorna un número real random distribuido
                    uniformemente dentro del rango [0,1).
   for k ← 1 to pop_size do
        sum ← sum + e_k;
        while (sum > ptr) do
             seleccionar cromosoma k;
             ptr ← ptr + 1;
        end while
   end for
end procedure
```

Figura 4.6 Procedimiento del Stochastic Universal Sampling

Un problema relacionado es que cuando se descartan los cromosomas duplicados, el tamaño de la población formada con los padres y los hijos puede ser menor que el tamaño predefinido *pop_size*. En este caso, se usa el procedimiento de iniciación para llenar el espacio vacante del pool de población.

4.4.2.2.MUESTREOS DETERMINÍSTICOS

Esta opción generalmente selecciona los mejores *pop_size* cromosomas desde el espacio de muestreo. Tanto la selección $(\mu + \lambda)$ como la (μ, λ) pertenecen este método. Ambas opciones prohiben que cromosomas duplicados entren en la población durante la selección.

Truncation selection y *block selection* pertenecen a este método, los cuales ordenan todos los cromosomas de acuerdo a su fitness y selecciona el mejor como padre [137]. En truncation selection se define un umbral *T* tal que se seleccionan los *T%* mejores cromosomas y cada uno recibe alrededor de 100/*T* copias. Block selection es equivalente a truncation selection ya que para una población dada de tamaño *pop_size*, se dan simplemente *s* copias a los *pop_size/s* mejores cromosomas.

La s*elección elitista* asegura que el mejor cromosoma se mantiene a través de las generaciones si no se selecciona a través del proceso de selección. Bajo ciertas condiciones muy generales, la introducción del elitismo garantiza la convergencia teórica al óptimo global; en la práctica, mejora la velocidad de convergencia de los GAs cuando la función de evaluación es unimodal, es decir, no existen subóptimos, sin embargo la velocidad de convergencia empeora con funciones fuertemente multimodales.

El muestreo determinístico de Brindle está basado sobre el concepto de *número esperado* [20]. La probabilidad de selección para cada cromosoma se calcula de la forma usual, $p_k = f_k / \sum f_k$. El número esperado de cada cromosoma se calcula como $e_k = p_k \times pop\_size$. A cada cromosoma se le asignan muestras en relación a la parte entera del número esperado, y entonces la población se ordena en función a la parte fraccional del número esperado. Los cromosomas restantes necesarios para completar se sacan de la parte superior de la lista ordenada.

El *reemplazo generacional* (reemplazo del conjunto entero de padres por sus hijos) se puede ver como otra versión de una opción determinística. Una modificación al

método es reemplazar los *n* cromosomas más viejos y peores con hijos (*n* es el número de hijos) [143, 102]. Los cromosomas a reemplazar por los hijos se seleccionan en función de su probabilidad de supervivencia. Los cromosomas peores en performance que la media tienen las chances más altas de ser seleccionados para morir.

4.4.2.3. MUESTREOS MIXTOS.

Esta opción tiene tanto características determinísticas como aleatorias. Un ejemplo típico es *selección por torneo* presentado por Goldberg [76]. Este método elige en forma aleatoria un conjunto de cromosomas y escoge el mejor del conjunto de reproducción. El número de cromosomas en el conjunto se llama *tamaño del torneo*. Un tamaño de torneo común es 2. Esto se denomina *torneo binario*.

Stochastic tournament selection fue sugerido por Wetzel [142]. En este método, las probabilidades de selección se calculan en la forma usual y se muestrean pares consecutivos de cromosomas usando selección por ruleta. Luego de sortear un par, esos cromosomas con alto fitness se insertan en la nueva población. El proceso continúa hasta que se completa la población.

Remainder stochastic sampling propuesto por Brindle es una versión modificada del muestreo determinístico [20]. En este método, a cada cromosoma se le asignan muestras acorde a la parte entera del número esperado, y entonces los cromosomas compiten en función de la parte fraccional del número esperado para los lugares restantes en la población.

4.4.3. PROBABILIDAD DE SELECCIÓN

En este punto lo más importante es determinar la probabilidad de selección de cada cromosoma. En el procedimiento de selección proporcional, la probabilidad de selección de un cromosoma es proporcional a su fitness. Este esquema simple exhibe algunas propiedades no deseables. Por ejemplo, en generaciones tempranas, hay una tendencia a que super individuos dominen el proceso de selección; en las generaciones finales, cuando la población ha convergido, la competencia entre cromosomas es menos fuerte y se producirá un comportamiento aleatorio de búsqueda donde la selección no concede mayor preeminencia a uno u otro individuo. En una población infinita los

procesos de muestreo son siempre imperfectos, pudiendo favorecer más de lo que le corresponde a individuos ocasionalmente más aptos; debido a que la población es finita y como consecuencia de esos errores estocásticos en los muestreos, un individuo puede acabar expulsando de la población a los restantes, haciendo que el GA converja prematuramente hacia tal individuo afortunado.

Para mitigar esos problemas se proponen los *mecanismos de escalamiento* y *ranking*. Los métodos de escalamiento transforman los valores crudos de la función objetivo a algún valor real positivo, y la probabilidad de supervivencia de cada cromosoma se determina en función de ese nuevo valor. Los métodos de ranking ignoran los valores de la función objetivo actual y en su lugar usan el ranking del cromosoma para determinar la probabilidad de supervivencia. El escalamiento del fitness tiene las siguientes intenciones:

1. Mantener una diferencia razonable entre los niveles de fitness relativo de los cromosomas.
2. Prevenir un copamiento muy rápido de algunos super cromosomas para reunir los requerimientos de limitar la competencia temprana y estimular la tardía.

Para la mayoría de los métodos de escalamiento, los parámetros de escalamiento son dependientes del problema. El ranking de fitness tiene un efecto similar al escalamiento de fitness, pero evita la necesidad de parámetros de escalamiento extras [116].

En general, el fitness escalado f'_k desde el fitness crudo (valor de la función objetivo) f'_k para el cromosoma k se puede expresar como [81]:

$$f'_k = g(f'_k)$$

donde la función $g(\cdot)$ transforma el fitness crudo en un fitness escalado. La función $g(\cdot)$ puede tomar diferentes formas para producir diferentes métodos de escalamiento, tal como escalamiento lineal, *sigma trunction* [75], *power law scaling* [73], *logarithmic scaling* [81], etc.

Baker introduce la noción de *ranking selection* para algoritmos genéticos [4, 76, 83, 54]. La idea es muy simple: ordenar la población del mejor al peor y asignar la probabilidad de selección de cada cromosoma acorde al ranking pero no ya a su fitness crudo. Hay dos métodos de uso común el *ranking lineal* y el *exponencial*.

4.4.4. PRESIÓN SELECTIVA

Para controlar la diversidad genética se debe actuar sobre el mecanismo de asignación de probabilidades de supervivencia (o de descendientes, en su caso): cuando más se favorezca a los más aptos menor variedad se obtendrá y, como consecuencia, se aumentará el riesgo de perder la información de valor desarrollada por los individuos moderadamente aptos, entendiendo por información de valor toda la que podría servir posteriormente para hallar el óptimo global. Por el contrario, si no se favorece especialmente a los individuos más aptos, se obtendrá más diversidad en la población, pero a costo de hacer las etapas de selección mucho menos eficientes, lo que empeora la eficiencia global del GA.

A la mayor o menor tendencia del mecanismo de asignación de probabilidades de supervivencia por favorecer a los individuos más aptos se le llama *presión selectiva*. Lo ideal sería que en las primeras fases de la evolución de un GA hubiera poca presión selectiva con el fin de que la búsqueda fuera lo más global posible y en las últimas fases de la evolución, una vez que ya sido localizada la mejor solución, se incrementará fuertemente para encontrar la mejor solución cuanto antes.

Desde el punto de vista Neo-Darwiniana, el proceso de evolución se puede dividir en tres categorías [99]:

- ✓ *Selección estabilizante.*
- ✓ *Selección direccional.*
- ✓ *Selección disruptiva.*

La *selección estabilizante* también se la llama como *selección normalizada*, porque tiende a eliminar cromosomas con valores extremos. La *selección direccional* tiene el efecto de incrementar o decrementar el valor medio de la población. La *selección disruptiva* tiende a eliminar cromosomas con valores moderados. La mayoría de los métodos de selección están basados en selección direccional.

4.5. OTROS COMPONENTES

4.5.1. ELECCIÓN DE LA POBLACIÓN INICIAL

En términos generales, la población inicial debe ser lo más variada posible. Es necesario que la distribución de aptitudes sea uniforme para evitar la convergencia

prematura. En la práctica, por falta de mayor información, se elige la población inicial al azar. El conocimiento específico puede ayudar a formar una población inicial factible con algún individuo cercano al óptimo.

La iniciación de la población en GA se realiza por muestreos aleatorios de una variable binaria, l . *pop_size* veces (muestreos independientes), donde l es la longitud del cromosoma; de esta forma se determina cada bit de la población inicial $P(0)$ en forma aleatoria.

4.5.2. CRITERIOS DE TERMINACIÓN

Los criterios de finalización de los GAs se identifican frecuentemente con un número máximo de generaciones. Sin embargo, algunas veces también se controla por la medida de la diversidad genética que permite calcular la convergencia promedio de todas las posiciones de bits en toda la población, es decir, la medida *bias* definida por Grefenstette [79]. Denotando un único individuo por $x_i=(x_{i,1},...,x_{i,l})$ ($\forall i \in \{1,\ldots, pop\_size\}$) para distinguir sus bits, el bias $b(P(t))$ de una población se calcula como:

$$b(P(t)) = \frac{1}{l \cdot pop\_size} \sum_{j=1}^{l} \max\left\{ \sum_{i=1}^{pop\_size} (1 - a_{i,j}), \sum_{i=1}^{pop\_size} (a_{i,j}) \right\} \in [0.5,1.0]$$

El valor $b \in [0.5, 1.0]$ indica el porcentaje promedio de los valores predominantes en cada posición de los individuos. Un valor pequeño de b indica una alta diversidad genotípica, mientras que un valor grande implica una baja diversidad.

4.6. LIMITACIONES DE LOS GAS E INCONVENIENTES ASOCIADOS

Las limitaciones principales de un GA básico se pueden describir de un modo general de la siguiente manera [116]:

- ✓ El espacio de búsqueda se puede representar solamente por cadenas binarias.
- ✓ El mecanismo de un algoritmo genético básico no considera las restricciones posibles que se pueden agregar a la búsqueda.

- ✓ El algoritmo genético básico es un algoritmo de búsqueda ciego, es decir, sólo está guiado por la aptitud de los individuos, y no incorpora ningún otro conocimiento específico del problema en cuestión.
- ✓ A efectos prácticos, el GA simple sólo puede trabajar con poblaciones finitas no muy grandes.

Los inconvenientes que producen las dos primeras limitaciones no se salen de lo previsible; mientras que las dos últimas son causa original de muchos inconvenientes que se deberán atacar por separado. De manera esquemática, se puede decir que todos esos inconvenientes tienen su base en dos hechos:

1. *Debilidad*.
2. *Problemas de diversidad* derivados del uso de poblaciones finitas.

4.6.1. EL PROBLEMA DE LA DEBILIDAD DE LOS GAS

El inconveniente más visible de la debilidad es la *ineficiencia*: los métodos débiles de resolución de problemas presentan las ventajas de ser muy poco específicos, poco exigentes y notablemente eficaces; sin embargo, todos ellos son ineficientes en mayor o menor grado, y especialmente se les compara con métodos heurísticos. Los GAs sólo se pueden considerar eficientes en comparación con otros métodos estocásticos de búsqueda ciega, pero se puede garantizar que si se encuentra un método heurístico para resolver un problema, este siempre lo hará mucho más eficientemente que un GA.

Otro de los inconvenientes prácticos de la debilidad es la tendencia al *extravío* de la búsqueda. El GA simple realiza la búsqueda de los mejores puntos utilizando únicamente la aptitud de los individuos para recombinar internamente los bloques constructivos; en ocasiones ocurre que la información proporcionada resulta insuficiente para orientar correctamente al algoritmo en su búsqueda del óptimo. A esto se le llama *desorientación*. En el ámbito interno esto se concreta con la no verificación de la hipótesis de los bloques constructivos, es decir, ciertas combinaciones válidas de buenos bloques constructivos originan individuos de baja aptitud. En el peor de los casos puede ocurrir que este fenómeno sea preponderante e impida que el GA converja al óptimo global, desviándolo finalmente hacia un óptimo local. Esto es el extravío propiamente dicho. Afortunadamente, aunque un GA se desoriente no necesariamente se va a

extraviar; para eso es necesario partir de una mala población inicial o plantear un problema especialmente diseñado para que se extravíe.

Para que el mecanismo básico de los GAs funcione correctamente es necesario proporcionarle una mínima cantidad y calidad de información. Si no se le proporciona ese mínimo el mecanismo no funciona con propiedad, y el GA evolucionará incorrectamente.

Como es de suponer, el factor que más contribuye a la existencia de desorientación es la forma de la función de fitness. Típicamente, la desorientación es frecuente en problemas en los que los puntos óptimos están aislados y rodeados de puntos de muy bajo fitness. Sin embargo, la desorientación suele estar fuertemente estimulada por la mala codificación que oculte la ya de por sí escasa información disponible. De modo recíproco, una buena codificación reduce la probabilidad de extravío.

Un caso especialmente desfavorable ocurre cuando hay una fuerte interacción entre dos o más atributos, de tal forma que la contribución a la aptitud de un individuo que realiza cierto gen depende grandemente de los valores que tomen otros. A este fenómeno se denomina *epistasis* o acoplamiento y su presencia garantiza la desorientación dado que en esas condiciones resulta muy difícil constituir buenos bloques constructivos.

4.6.2. EL PROBLEMA DE LA DIVERSIDAD EN LOS GAS

Es esencial para el buen funcionamiento de un GA tener controlada en todo momento la diversidad de la población. En un sentido general se entiende por diversidad a la variedad de individuos y en particular a la variedad de aptitudes.

La necesidad de la diversidad de individuos se debe a que con poca variedad el operador de crossover pierde casi por completo la capacidad de intercambio de información útil entre individuos, y en definitiva, la búsqueda se estanca. La necesidad de tener controlada la diversidad de aptitudes (de individuos) radica en la imposibilidad práctica de trabajar con una población infinita.

Con poca diversidad de aptitudes, todos los individuos tendrán más o menos las mismas posibilidades de sobrevivir, la selección no incrementará la diversidad por lo que el peso de la búsqueda recaerá en los operadores genéticos, lo que traerá como

consecuencia una búsqueda aleatoria. Para que la selección resulte efectiva, la población debe contener en todo momento una cierta variedad de aptitudes.

Cuando la población es finita tampoco se puede tener gran disparidad de aptitudes, pues ello suele afectar muy negativamente a la diversidad de la población.

Es imprescindible tener control sobre la diversidad de aptitudes de la población para evitar que se produzca una convergencia prematura ya sea por mucha diversidad (superindividuos que son aptos en un cierto momento pero no los más aptos absolutos) o incluso por demasiado poca (deriva genética).