Tema 2. Algoritmos Genéticos

Abdelmalik Moujahid, Iñaki Inza y Pedro Larrañaga
Departamento de Ciencias de la Computación e Inteligencia Artificial
Universidad del País Vasco–Euskal Herriko Unibertsitatea

2.1 Introducción

Los Algoritmos Genéticos (AGs) son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. Están basados en el proceso genético de los organismos vivos. A lo largo de las generaciones, las poblaciones evolucionan en la naturaleza de acorde con los principios de la selección natural y la supervivencia de los más fuertes, postulados por Darwin (1859). Por imitación de este proceso, los Algoritmos Genéticos son capaces de ir creando soluciones para problemas del mundo real. La evolución de dichas soluciones hacia valores óptimos del problema depende en buena medida de una adecuada codificación de las mismas.

Los principios básicos de los Algoritmos Genéticos fueron establecidos por Holland (1975), y se encuentran bien descritos en varios textos – Goldberg (1989), Davis (1991), Michalewicz (1992), Reeves (1993) – .

En la naturaleza los individuos de una población compiten entre sí en la búsqueda de recursos tales como comida, agua y refugio. Incluso los miembros de una misma especie compiten a menudo en la búsqueda de un compañero. Aquellos individuos que tienen más éxito en sobrevivir y en atraer compañeros tienen mayor probabilidad de generar un gran número de descendientes. Por el contrario individuos poco dotados producirán un menor número de descendientes. Esto significa que los genes de los individuos mejor adaptados se propagarán en sucesivas generaciones hacia un número de individuos creciente. La combinación de buenas características provenientes de diferentes ancestros, puede a veces producir descendientes "superindividuos", cuya adaptación es mucho mayor que la de cualquiera de sus ancestros. De esta manera, las especies evolucionan logrando unas características cada vez mejor adaptadas al entorno en el que viven.

Los Algoritmos Genéticos usan una analogía directa con el comportamiento natural. Trabajan con una población de individuos, cada uno de los cuales representa una solución factible a un problema dado. A cada individuo se le asigna un valor ó puntuación, relacionado con la bondad de dicha solución. En la naturaleza esto equivaldría al grado de efectividad de un organismo para competir por unos determinados recursos. Cuanto mayor sea la adaptación de un individuo al problema, mayor será la probabilidad de que el mismo sea seleccionado para reproducirse, cruzando su material genético con otro individuo seleccionado de igual forma. Este cruce producirá nuevos individuos – descendientes de los anteriores – los cuales comparten algunas de las características de sus padres. Cuanto menor sea la adaptación de un individuo, menor será la probabilidad de que dicho individuo sea seleccionado para la reproducción, y por tanto de que su material genético se propague en sucesivas generaciones.
De esta manera se produce una nueva población de posibles soluciones, la cual reem-

plaza a la anterior y verifica la interesante propiedad de que contiene una mayor proporción de buenas características en comparación con la población anterior. Así a lo largo de las generaciones las buenas carácterísticas se propagan a travées de la población. Favoreciendo el cruce de los individuos mejor adaptados, van siendo exploradas las áreas más prometedoras del espacio de búsqueda. Si el Algoritmo Genético ha sido bien diseñado, la población convergerá hacia una solución óptima del problema.

El poder de los Algoritmos Genéticos proviene del hecho de que se trata de una técnica robusta, y pueden tratar con éxito una gran variedad de problemas provenientes de diferentes áreas, incluyendo aquellos en los que otros métodos encuentran dificultades. Si bien no se garantiza que el Algoritmo Genético encuentre la solución óptima del problema, existe evidencia empírica de que se encuentran soluciones de un nivel aceptable, en un tiempo competitivo con el resto de algoritmos de optimización combinatoria. En el caso de que existan técnicas especializadas para resolver un determinado problema, lo más probable es que superen al Algoritmo Genético, tanto en rapidez como en eficacia. El gran campo de aplicación de los Algoritmos Genéticos se relaciona con aquellos problemas para los cuales no existen técnicas especializadas. Incluso en el caso en que dichas técnicas existan, y funcionen bien, pueden efectuarse mejoras de las mismas hibridándolas con los Algoritmos Genéticos.

La estructura de este tema es como sigue: en la siguiente sección se introduce por medio de un ejemplo el denominado Algoritmo Genético Simple, también conocido como Algoritmo Genético Canónico, para a continuación, mostrar distintas extensiones y modificaciones del mismo, relativas a los operadores de selección, cruce, mutación y reducción, así como a la hibridación del Algoritmo Genético con otros algoritmos de búsqueda local, y a diversos modelos de Algoritmos Genéticos Distribuidos. En la siguiente sección nos preguntamos el motivo por el cual funcionan los Algoritmos Genéticos, demostrándose el teorema de los esquemas, y referenciándose algunos trabajos teóricos relacionados con las condiciones suficientes para garantizar la convergencia de dichos algoritmos hacia el óptimo global. Finalizamos el capítulo, mostrando operadores de cruce y mutación específicos para el problema del agente viajero.

2.2 El Algoritmo Genético Simple

El Algoritmo Genético Simple, también denominado Canónico, se representa en la figura 2.1. Como se verá a continuación, se necesita una codificación o representación del problema, que resulte adecuada al mismo. Además se requiere una función de ajuste ó adaptación al problema, la cual asigna un número real a cada posible solución codificada. Durante la ejecución del algoritmo, los padres deben ser seleccionados para la reproducción, a continuación dichos padres seleccionados se cruzarán generando dos hijos, sobre cada uno de los cuales actuará un operador de mutación. El resultado de la combinación de las anteriores funciones será un conjunto de individuos (posibles soluciones al problema), los cuales en la evolución del Algoritmo Genético formarán parte de la siguiente población.

2.2.1 Codificación

Se supone que los individuos (posibles soluciones del problema), pueden representarse como un conjunto de parámetros (que denominaremos *genes*), los cuales agrupados forman una ristra de valores (a menudo referida como *cromosoma*). Si bien el alfabeto

```
BEGIN /* Algoritmo Genetico Simple */
    Generar una poblacion inicial.
    Computar la funcion de evaluacion de cada individuo.
    WHILE NOT Terminado DO
    BEGIN /* Producir nueva generacion */
        FOR Tamaño poblacion/2 DO
        BEGIN /*Ciclo Reproductivo */
            Seleccionar dos individuos de la anterior generacion,
            para el cruce (probabilidad de seleccion proporcional
            a la funcion de evaluacion del individuo).
            Cruzar con cierta probabilidad los dos
            individuos obteniendo dos descendientes.
            Mutar los dos descendientes con cierta probabilidad.
            Computar la funcion de evaluacion de los dos
            descendientes mutados.
            Insertar los dos descendientes mutados en la nueva generacion.
        END
        IF  la poblacion ha convergido THEN
            Terminado := TRUE
    END
END
```

Figura 2.1: Pseudocódigo del Algoritmo Genético Simple

utilizado para representar los individuos no debe necesariamente estar constituido por el $\{0, 1\}$, buena parte de la teoría en la que se fundamentan los Algoritmos Genéticos utiliza dicho alfabeto.

En términos biológicos, el conjunto de parámetros representando un cromosoma particular se denomina *fenotipo*. El fenotipo contiene la información requerida para construir un organismo, el cual se refiere como *genotipo*. Los mismos términos se utilizan en el campo de los Algoritmos Genéticos. La adaptación al problema de un individuo depende de la evaluación del genotipo. Esta última puede inferirse a partir del fenotipo, es decir puede ser computada a partir del cromosoma, usando la función de evaluación.

La *función de adaptación* debe ser diseñada para cada problema de manera específica. Dado un cromosoma particular, la función de adaptación le asigna un número real, que se supone refleja el nivel de adaptación al problema del individuo representado por el cromosoma.

Durante la *fase reproductiva* se seleccionan los individuos de la población para cruzarse y producir descendientes, que constituirán, una vez mutados, la siguiente generación de individuos. La *selección de padres* se efectúa al azar usando un procedimiento que favorezca a los individuos mejor adaptados, ya que a cada individuo se le asigna una probabilidad de ser seleccionado que es proporcional a su función de adaptación. Este procedimiento se dice que está basado en la ruleta sesgada. Según dicho esquema, los individuos bien adaptados se escogerán probablemente varias veces por generación, mientras que los pobremente adaptados al problema, no se escogerán más que de vez en cuando.

Una vez seleccionados dos padres, sus cromosomas se combinan, utilizando habitualmente los *operadores* de *cruce* y *mutación*. Las formas básicas de dichos operadores

se describen a continuación.

El *operador de cruce*, coge dos padres seleccionados y corta sus ristras de cromosomas en una posición escogida al azar, para producir dos subristras iniciales y dos subristras finales. Después se intercambian las subristras finales, produciéndose dos nuevos cromosomas completos (véase la Figura 2.2). Ambos descendientes heredan genes de cada uno de los padres. Este operador se conoce como operador de cruce basado en un punto. Habitualmente el operador de cruce no se aplica a todos los pares de in-

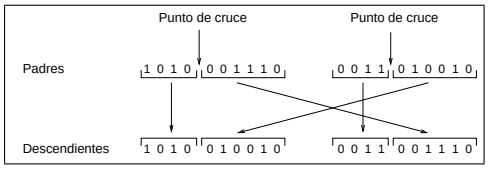

Figura 2.2: Operador de cruce basado en un punto

dividuos que han sido seleccionados para emparejarse, sino que se aplica de manera aleatoria, normalmente con una probabilidad comprendida entre 0.5 y 1.0. En el caso en que el operador de cruce no se aplique, la descendencia se obtiene simplemente duplicando los padres.

El *operador de mutación* se aplica a cada hijo de manera individual, y consiste en la alteración aleatoria (normalmente con probabilidad pequeña) de cada gen componente del cromosoma. La Figura 2.3 muestra la mutación del quinto gen del cromosoma. Si bien puede en principio pensarse que el operador de cruce es más importante

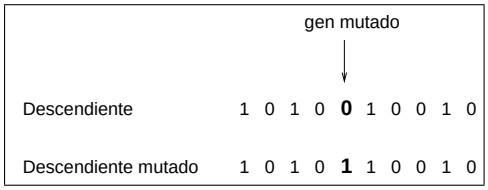

Figura 2.3: Operador de mutación

que el operador de mutación, ya que proporciona una exploración rápida del espacio de búsqueda, éste último asegura que ningún punto del espacio de búsqueda tenga probabilidad cero de ser examinado, y es de capital importancia para asegurar la convergencia de los Algoritmos Genéticos.

Para criterios prácticos, es muy útil la definición de convergencia introducida en este campo por De Jong (1975) en su tesis doctoral. Si el Algoritmo Genético ha sido correctamente implementado, la población evolucionará a lo largo de las generaciones sucesivas de tal manera que la adaptación media extendida a todos los individuos de la población, así como la adaptación del mejor individuo se irán incrementando hacia el óptimo global. El concepto de convergencia está relacionado con la progresión hacia la uniformidad: un gen ha convergido cuando al menos el 95 % de los individuos de la población comparten el mismo valor para dicho gen. Se dice que la población converge cuando todos los genes han convergido. Se puede generalizar dicha definición al caso en que al menos un $\beta\,\%$ de los individuos de la población hayan convergido.

La Figura 2.4 muestra como varía la adaptación media y la mejor adaptación en un Algoritmo Genético Simple típico. A medida que el número de generaciones aumenta, es más probable que la adaptación media se aproxime a la del mejor individuo.

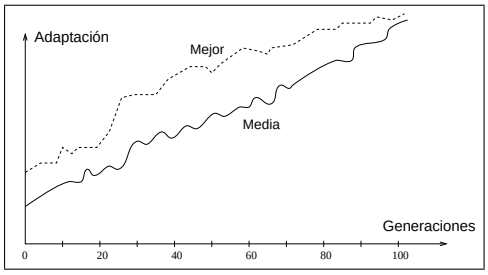

Figura 2.4: Adaptación media y mejor adaptación en un Algoritmo Genético Simple

	Población inicial (fenotipos)	x valor genotipo	$f(x)$ valor (función adaptación)	$f(x)/\sum f(x)$ (probabilidad selección)	Probabilidad de selección acumulada
1	01101	13	169	0.14	0.14
2	11000	24	576	0.49	0.63
3	01000	8	64	0.06	0.69
4	10011	19	361	0.31	1.00
Suma			1170		
Media			293		
Mejor			576		

Tabla 2.1: Población inicial de la simulación efectuada a mano correspondiente al Algoritmo Genético Simple

2.2.2 Ejemplo

Como ilustración de los diferentes componentes del Algoritmo Genético Simple, supongamos el problema – adaptado de Goldberg (1989) – de encontrar el máximo de la función $f(x) = x^2$ sobre los enteros $\{1, 2, \ldots, 32\}$. Evidentemente para lograr dicho óptimo, bastaría actuar por búsqueda exhaustiva, dada la baja cardinalidad del espacio de búsqueda. Se trata por tanto de un mero ejemplo con el que pretendemos ilustrar el comportamiento del algoritmo anteriormente descrito. Consultando el pseudocódigo de la Figura 2.1, vemos que el primer paso a efectuar consiste en determinar el tamaño de la población inicial, para a continuación obtener dicha población al azar y computar la función de evaluación de cada uno de sus individuos.

Suponiendo que el alfabeto utilizado para codificar los individuos esté constituido por $\{0, 1\}$, necesitaremos ristras de longitud 5 para representar los 32 puntos del espacio de búsqueda.

En la Tabla 2.1, hemos representado los 4 individuos que constituyen la población inicial, junto con su función de adaptación al problema, así como la probabilidad de que cada uno de dichos individuos sea seleccionado – según el modelo de ruleta sesgada – para emparejarse.

Volviendo a consultar el pseudocódigo expresado en la Figura 2.1, vemos que el siguiente paso consiste en la selección de 2 parejas de individuos. Para ello es suficiente, con obtener 4 números reales provenientes de una distribución de probabilidad uniforme en el intervalo $[0, 1]$, y compararlos con la última columna de la Tabla 2.1. Así por ejemplo, supongamos que dichos 4 números hayan sido: 0.58; 0.84; 0.11 y 0.43. Esto significa que los individuos seleccionados para el cruce han sido: el individuo 2 junto con el individuo 4, así como el individuo 1 junto con el individuo 2.

Para seguir con el Algoritmo Genético Simple, necesitamos determinar la probabilidad de cruce, p_c. Supongamos que se fije en $p_c = 0{,}8$. Valiéndonos al igual que antes

Emparejamiento de los individuos seleccionados	Punto de cruce	Descendientes	Nueva población descendientes mutados	x valor genotipo	$f(x)$ función adaptación
11000	2	11011	11011	27	729
10011	2	10000	10000	16	256
01101	3	01100	11100	28	784
11000	3	11101	11101	29	841
Suma					2610
Media					652.5
Mejor					841

Tabla 2.2: Población en el tiempo 1, proveniente de efectuar los operadores de cruce y mutación sobre los individuos expresados en la Tabla 2.1, los cuales constituyen la población en el tiempo 0

de, 2 en este caso, números provenientes de la distribución uniforme, determinaremos si los emparejamientos anteriores se llevan a cabo. Admitamos, por ejemplo, que los dos números extraídos sean menores que 0.8, decidiéndose por tanto efectuar el cruce entre las dos parejas. Para ello escogeremos un número al azar entre 1 y $l-1$ (siendo l la longitud de la ristra utilizada para representar el individuo). Notése que la restricción impuesta al escoger el número entre 1 y $l-1$, y no l, se realiza con la finalidad de que los descendientes no coincidan con los padres.

Supongamos, tal y como se indica en la Tabla 2.2, que los puntos de cruce resulten ser 2 y 3. De esta manera obtendríamos los 4 descendientes descritos en la tercera columna de la Tabla 2.2. A continuación siguiendo el pseudocódigo de la Figura 2.1, mutaríamos con una probabilidad, p_m, cercana a cero, cada uno de los bit de las cuatro ristras de individuos. En este caso suponemos que el único bit mutado corresponde al primer gen del tercer individuo. En las dos últimas columnas se pueden consultar los valores de los individuos, así como las funciones de adaptación correspondientes. Como puede observarse, tanto el mejor individuo como la función de adaptación media han mejorado sustancialmente al compararlos con los resultados de la Tabla 2.1.

3. Extensiones y Modificaciones del AGS

En este apartado se introducirán algunas extensiones y modificaciones del Algoritmo Genético Simple. Se comenzará dando un pseudocódigo para un Algoritmo Genético Abstracto, para a continuación formalizar matemáticamente cada uno de los elemen-

```
BEGIN AGA
    Obtener la poblacion inicial al azar.
    WHILE NOT stop DO
        BEGIN
            Seleccionar padres de la poblacion.
            Producir hijos a partir de los padres seleccionados.
            Mutar los individuos hijos.
            Extender la poblacion añadiendo los hijos.
            Reducir la poblacion extendida.
        END
END AGA
```

Figura 2.5: Pseudocódigo del Algoritmo Genético Abstracto

tos integrantes del Algoritmo, así como las extensiones y modificaciones que se vayan presentando.

Una versión del Algoritmo Genético Abstracto (AGA), puede ser como el de la Figura 2.5. Ya que el principal dominio de aplicación de los Algoritmos Genéticos lo constituye la optimización de funciones, se introducen algunos conceptos básicos que se usarán a lo largo de este tema.

Dado un dominio finito D y una función $f : D \to \Re$, el problema de la *optimización de la función* f se refiere a encontrar el mejor valor de la función f en el dominio D. Se trata por tanto de encontrar $x \in D$ para el cual $f(x) \leq f(y)\ \forall y \in D$.

Ya que $\text{máx}\{f(x)\} = -\,\text{mín}\{-f(x)\}$ la restricción al problema de minimización no supone ninguna pérdida de generalización. En general, la tarea de optimización se complica debido a la existencia de óptimos locales (mínimos locales en nuestro caso).

La función f tiene un mínimo local en $\hat{x} \in D$ si se verifica la siguiente condición:

$$\exists E(x), \text{ entorno de } x, \text{ tal que si } y \in E(x), \Rightarrow f(x) \leq f(y).$$

Diremos que $e : D \to S^l$ donde $l \geq \log_{\|S\|} \|D\|$ constituye una *codificación*, siendo la finalidad de la misma el representar los elementos de D por medio de ristras de elementos de S. A S se le denomina *alfabeto*, mientras que S^l constituye el *espacio de búsqueda*. A la función $f(x) = g(e(x))$ se le denomina *función objetivo*. Es necesario que la función e sea inyectiva, para que los elementos de D sean discernibles.

El AGA examinará un subconjunto del espacio de búsqueda, obteniendo una ristra x^*, cuya función objetivo $g(x^*)$ puede considerarse un estimador del $\text{mín}_{x \in S^l}\, g(x)$.

Abusando del lenguaje de notación, designaremos por I a los elementos de S^l.

En lo que sigue se considerará un AGA como una 10–tupla:

$$\text{AGA} = (P_0, \lambda, l, f_{\text{sel}}, f_{\text{prod}}, f_{\text{mut}}, f_{\text{ext}}, f_{\text{red}}, g, ct),$$

donde:

$P_0 = \{I_0^1, \ldots, I_0^\lambda\} \in (S^l)^\lambda$	población inicial,
λ	tamaño población,
l	longitud de la representación,
f_{sel}	función de selección,
f_{prod}	función de producción de hijos,
f_{mut}	función de mutación,
f_{ext}	función de extensión,
f_{red}	función de reducción,
g	función objetivo,
ct	criterio de parada.

En principio restringiremos nuestro AGA, imponiéndole la condición de que todas las poblaciones tengan el mismo tamaño. Obviamente una generalización sería el considerar que el tamaño depende de la generación, es decir $\lambda_t =\mid P_t \mid$. Denotaremos por $\mathcal{P}_\lambda$ el conjunto de poblaciones de tamaño λ, a las que denominaremos *poblaciones bien dimensionadas*.

La *función de selección global*, f_{sel}, selecciona al azar y con reemplazamiento una colección de individuos $y \in \mathcal{P}_\lambda$ a partir de una población $x \in \mathcal{P}_\lambda$:

$$f_{sel} : (\alpha, x) \longrightarrow y,$$

donde α es un vector de dimensión λ constituído por valores escogidos aleatoriamente.

La *función de reproducción global*, f_{prod}, produce una población de descendientes $z \in \mathcal{P}_\lambda$ a partir de individuos seleccionados $y \in \mathcal{P}_\lambda$ por medio de un operador de cruce:

$$f_{prod} : (\beta, y) \longrightarrow z,$$

donde β es un vector de dimensión $\lambda/2$ de valores escogidos al azar entre los enteros $\{1, ..., l-1\}$

La *función de reproducción* se dice que está *basada en un punto* si los padres $I^i = (s_1, ..., s_l)$ y $I^j = (b_1, ..., b_l)$ producen hijos $CH^{i,j;1} = (c_1, ..., c_l)$ y $CH^{i,j;2} = (d_1, ..., d_l)$ verificándose :

$$c_j = \begin{cases} s_j & \text{si } j \leq m \\ b_j & \text{si } j > m \end{cases} \tag{1}$$

$$d_j = \begin{cases} b_j & \text{si } j \leq m \\ s_j & \text{si } j > m \end{cases} \tag{2}$$

donde m es un número entero escogido al azar según una distribución uniforme discreta definida sobre el conjunto $\{1, ..., l-1\}$.

La *función de mutación individual* $f_{ind-mut}$, aplicada a $I = (s_1, \ldots, s_l)$, genera otro individuo $MI = (sm_1, \ldots, sm_l)$, es decir $f_{ind-mut}(I) = MI$, tal que $\forall j \in \{1, \ldots, l\}$, $P(sm_j = s_j) = 1 - p_m$, donde p_m es la probabilidad de mutación.

La *función de extensión*, f_{ext}, crea a partir de dos poblaciones $x, z \in \mathcal{P}_\lambda$, una población $n \in \mathcal{P}_{2\lambda}$:

$$f_{ext} : (x, z) \longrightarrow n$$

Denotando por N_i con $i = 1, .., 2\lambda$ el i-ésimo individuo en n, por X_k, con $k = 1, ..., \lambda$ el k-ésimo individuo en x, y por Z_j con $j = 1, ..., \lambda$ el j-ésimo individuo en z, se tendrá:

$$N_i = \begin{cases} X_i & \text{si } i \leq \lambda \\ Z_{i-\lambda} & \text{si } i > \lambda \end{cases} \tag{3}$$

La *función de reducción global*, f_{red}, convierte una población $n \in \mathcal{P}_{2\lambda}$ en una población $r \in \mathcal{P}_\lambda$

$$f_{red} : n \longrightarrow r.$$

Nótese que r denota la población de individuos en el tiempo $t + 1$.

La *función de reducción* es *elitista de grado* λ si la población en el tiempo $t + 1$ está constituida por los mejores λ individuos, de entre los λ individuos que constituyen la población en el tiempo t y los descendientes derivados de ellos.

La *función de reducción* se denomina *simple* si la población en el tiempo $t+1$ está formada por los descendientes derivados de la población en el tiempo t.

2.3.1 Población

Tamaño de la población

Una cuestión que uno puede plantearse es la relacionada con el tamaño idóneo de la población. Parece intuitivo que las poblaciones pequeñas corren el riesgo de no cubrir adecuadamente el espacio de búsqueda, mientras que el trabajar con poblaciones de gran tamaño puede acarrear problemas relacionados con el excesivo costo computacional.

Goldberg (1989) efectuó un estudio teórico, obteniendo como conclusión que el tamaño óptimo de la población para ristras de longitud l, con codificación binaria, crece exponencialmente con el tamaño de la ristra.

Este resultado traería como consecuencia que la aplicabilidad de los Algoritmos Genéticos en problemas reales sería muy limitada, ya que resultarían no competitivos con otros métodos de optimización combinatoria. Alander (1992), basándose en evidencia empírica sugiere que un tamaño de población comprendida entre l y $2l$ es suficiente para atacar con éxito los problemas por el considerados.

Población inicial

Habitualmente la población inicial se escoge generando ristras al azar, pudiendo contener cada gen uno de los posibles valores del alfabeto con probabilidad uniforme. Nos podríamos preguntar que es lo que sucedería si los individuos de la población inicial se obtuviesen como resultado de alguna técnica heurística o de optimización local. En los pocos trabajos que existen sobre este aspecto, se constata que esta inicialización no aleatoria de la población inicial, puede acelerar la convergencia del Algoritmo Genético. Sin embargo en algunos casos la desventaja resulta ser la prematura convergencia del algoritmo, queriendo indicar con ésto la convergencia hacia óptimos locales.

2.3.2 Función objetivo

Dos aspectos que resultan cruciales en el comportamiento de los Algoritmos Genéticos son la determinación de una adecuada función de adaptación o función objetivo, así como la codificación utilizada.

Idealmente nos interesaría construir funciones objetivo con "ciertas regularidades", es decir funciones objetivo que verifiquen que para dos individuos que se encuentren cercanos en el espacio de búsqueda, sus respectivos valores en las funciones objetivo sean similares. Por otra parte una dificultad en el comportamiento del Algoritmo Genético puede ser la existencia de gran cantidad de óptimos locales, así como el hecho de que el óptimo global se encuentre muy aislado.

La regla general para construir una buena función objetivo es que ésta debe reflejar el valor del individuo de una manera "real", pero en muchos problemas de optimización combinatoria, donde existen gran cantidad de restricciones, buena parte de los puntos del espacio de búsqueda representan individuos no válidos.

Para este planteamiento en el que los individuos están sometidos a restricciones, se han propuesto varias soluciones. La primera sería la que podríamos denominar *absolutista*, en la que aquellos individuos que no verifican las restricciones, no son considerados como tales, y se siguen efectuando cruces y mutaciones hasta obtener individuos válidos, o bien a dichos individuos se les asigna una función objetivo igual a cero.

Otra posibilidad consiste en reconstruir aquellos individuos que no verifican las restricciones. Dicha reconstrucción suele llevarse a cabo por medio de un nuevo operador que se acostumbra a denominar *reparador*.

Otro enfoque está basado en la *penalización de la función objetivo*. La idea general consiste en dividir la función objetivo del individuo por una cantidad (la penalización) que guarda relación con las restricciones que dicho individuo viola. Dicha cantidad puede simplemente tener en cuenta el número de restricciones violadas ó bien el denominado *costo esperado de reconstrucción*, es decir el coste asociado a la conversión de dicho individuo en otro que no viole ninguna restricción.

Otra técnica que se ha venido utilizando en el caso en que la computación de la función objetivo sea muy compleja es la denominada *evaluación aproximada de la función objetivo*. En algunos casos la obtención de n funciones objetivo aproximadas puede resultar mejor que la evaluación exacta de una única función objetivo (supuesto el caso de que la evaluación aproximada resulta como mínimo n veces más rápida que la evaluación exacta).

Un problema habitual en las ejecuciones de los Algoritmos Genéticos surge debido a la velocidad con la que el algoritmo converge. En algunos casos la convergencia es muy rápida, lo que suele denominarse *convergencia prematura*, en la cual el algoritmo converge hacia óptimos locales, mientras que en otros casos el problema es justo el contrario, es decir se produce una *convergencia lenta* del algoritmo. Una posible solución a estos problemas pasa por efectuar transformaciones en la función objetivo. El problema de la convergencia prematura, surge a menudo cuando la selección de individuos se realiza de manera proporcional a su función objetivo. En tal caso, pueden existir individuos con una adaptación al problema muy superior al resto, que a medida que avanza el algoritmo "dominan" a la población. Por medio de una transformación de la función objetivo, en este caso una comprensión del rango de variación de la función objetivo, se pretende que dichos "superindividuos" no lleguen a dominar a la población.

El problema de la lenta convergencia del algoritmo, se resolvería de manera análoga, pero en este caso efectuando una expansión del rango de la función objetivo.

La idea de especies de organismos, ha sido imitada en el diseño de los Algoritmos Genéticos en un método propuesto por Goldberg y Richardson (1987), utilizando una *modificación de la función objetivo* de cada individuo, de tal manera que individuos que estén muy cercanos entre sí devalúen su función objetivo, con objeto de que la población gane en diversidad.

Por ejemplo, si denotamos por $d(I_t^j, I_t^i)$ a la distancia de Hamming entre los individuos I_t^j e I_t^i, y por $K \in \mathcal{R}^+$ a un parámetro, podemos definir la siguiente función:

$$h(d(I_t^j, I_t^i)) = \begin{cases} K - d(I_t^j, I_t^i) & \text{si } d(I_t^j, I_t^i) < K, \\ 0 & \text{si } d(I_t^j, I_t^i) \geq K. \end{cases}$$

A continuación para cada individuo I_t^j, definimos $\sigma_j^t = \sum_{i \neq j} h(d(I_t^j, I_t^i))$, valor que utilizaremos para devaluar la función objetivo del individuo en cuestión. Es decir, $g^*(I_t^j) = g(I_t^j)/\sigma_j^t$. De esta manera aquellos individuos que están cercanos entre sí verán devaluada la probabilidad de ser seleccionados como padres, aumentándose la probabilidad de los individuos que se encuentran más aislados.

2.3.3 Selección

La función de selección de padres más utilizada es la denominada *función de*

selección proporcional a la función objetivo, en la cual cada individuo tiene una probabilidad de ser seleccionado como padre que es proporcional al valor de su función objetivo.

Denotando por $p_{j,t}^{\text{prop}}$ la probabilidad de que el individuo I_t^j sea seleccionado como padre, se tiene que:

$$p_{j,t}^{\text{prop}} = \frac{g(I_t^j)}{\sum_{j=1}^{\lambda} g(I_t^j)}.$$

Esta función de selección es invariante ante un cambio de escala, pero no ante una traslación.

Una de las maneras de superar el problema relacionado con la rápida convergencia proveniente de los superindividuos, que surge al aplicar la anterior función de selección, es el efectuar la *selección proporcional al rango del individuo*, con lo cual se produce una repartición más uniforme de la probabilidad de selección, tal y como se ilustra en la Figura 2.6. Si denotamos por $\text{rango}(g(I_t^j))$ el rango de la función objetivo

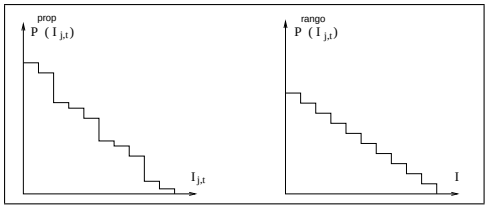

Figura 2.6: Esquemas de selección de padres proporcional a la función objetivo (izquierda) y proporcional al rango de la función objetivo (derecha)

del individuo I_t^j cuando los individuos de la población han sido ordenados de menor a mayor (es decir el peor individuo tiene rango 1, mientras que el individuo con mejor función objetivo tiene rango λ), y sea $p_{j,t}^{\text{rango}}$ la probabilidad de que el individuo I_t^j sea seleccionado como padre cuando la selección se efectúa proporcionalmente al rango del individuo, se tiene que

$$p_{j,t}^{\text{rango}} = \frac{\text{rango}(g(I_t^j))}{\lambda(\lambda+1)/2}.$$

La suma de los rangos, $\lambda(\lambda+1)/2$, constituye la constante de normalización. La función de selección basada en el rango es invariante frente a la translación y al cambio de escala.

Otro posible refinamento del modelo de selección proporcional, es el *modelo de selección del valor esperado*, el cual actúa de la manera siguiente: para cada individuo I_t^j, se introduce un contador, inicializado en $g(I_t^j)/\bar{g}_t$, donde $\bar{g}_t$ denota la media de la función objetivo en la generación t. Cada vez que el individuo I_t^j es seleccionado para el cruce, dicho contador decrece en una cantidad c ($c \in (0,5;1)$). El individuo en cuestión dejará de poder ser seleccionado en esa generación, cuando su contador sea negativo.

Baker (1987) introduce un método denominado *muestreo universal estocástico*, el cual utiliza un único giro de la ruleta siendo los sectores circulares proporcionales a la función objetivo. Los individuos son seleccionados a partir de marcadores (véase Figura 2.7), igualmente espaciados y con comienzo aleatorio.

Efectuando un paralelismo con los métodos de muestreo estadísticos, este último tipo

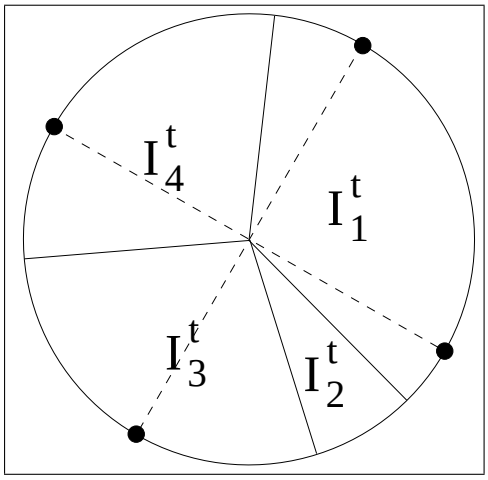

Figura 2.7: Método de selección de padres denominado muestreo universal estocástico. El individuo I_1^t se escoge 2 veces, mientras que I_3^t e I_4^t son elegidos una única vez

de selección de padres se relaciona con el muestreo sistemático, mientras que la selección proporcional a la función objetivo, está basada en el muestreo estratificado con afijación proporcional al tamaño.

En el modelo de *selección elitista* se fuerza a que el mejor individuo de la población en el tiempo t, sea seleccionado como padre.

La *selección por torneo*, constituye un procedimiento de selección de padres muy extendido y en el cual la idea consiste en escoger al azar un número de individuos de la población, tamaño del torneo, (con o sin reemplazamiento), seleccionar el mejor individuo de este grupo, y repetir el proceso hasta que el número de individuos seleccionados coincida con el tamaño de la población. Habitualmente el tamaño del torneo es 2, y en tal caso se ha utilizado una versión probabilística en la cual se permite la selección de individuos sin que necesariamente sean los mejores.

Una posible clasificación de procedimientos de selección de padres consistirá en: *métodos de selección dinámicos*, en los cuales los probabilidades de selección varían de generación a generación, (por ejemplo la selección proporcional a la función objetivo), frente a *métodos de selección estáticos*, en los cuales dichas probabilidades permanecen constantes (por ejemplo la selección basada en rangos).

Si se asegura que todos los individuos tienen asignada una probabilidad de selección distinta de cero el método de selección se denomina *preservativo*. En caso contrario se acostumbra a denominarlo *extintivo*.

2.3.4 Cruce

El Algoritmo Genético Canónico descrito anteriormente, utiliza el *cruce basado en un punto*, en el cual los dos individuos seleccionados para jugar el papel de padres, son recombinados por medio de la selección de un punto de corte, para posteriormente intercambiar las secciones que se encuentran a la derecha de dicho punto.

Se han investigado otros operadores de cruce, habitualmente teniendo en cuenta más de un punto de cruce. De Jong (1975) investigó el comportamiento del *operador de cruce basado en múltiples puntos*, concluyendo que el cruce basado en dos puntos,

representaba una mejora mientras que añadir más puntos de cruce no beneficiaba el comportamiento del algoritmo. La ventaja de tener más de un punto de cruce radica en que el espacio de búsqueda puede ser explorado más fácilmente, siendo la principal desventaja el hecho de aumentar la probabilidad de ruptura de buenos esquemas.

En el *operador de cruce basado en dos puntos*, los cromosomas (individuos) pueden contemplarse como un circuito en el cual se efectúa la selección aleatoria de dos puntos, tal y como se indica en la Figura 2.8.

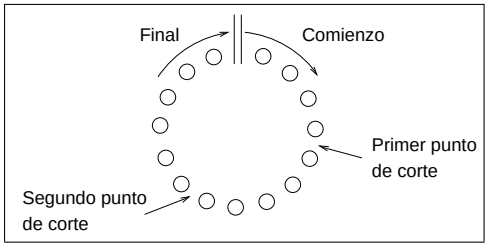

Figura 2.8: Individuo visto como un circuito

Desde este punto de vista, el cruce basado en un punto, puede verse como un caso particular del cruce basado en dos puntos, en el cual uno de los puntos de corte se encuentra fijo al comienzo de la ristra que representa al individuo. Véase Figura 2.9.

En el denominado *operador de cruce uniforme* (Syswerda (1991)) cada gen en la des-

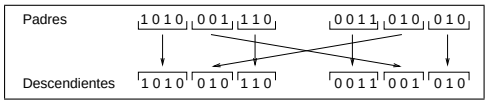

Figura 2.9: Operador de cruce basado en dos puntos

cendencia se crea copiando el correspondiente gen de uno de los dos padres, escogido de acuerdo a una "máscara de cruce" generada aleatoriamente. Cuando existe un 1 en la "máscara de cruce", el gen es copiado del primer padre, mientras que cuando exista un 0 en la máscara, el gen se copia del segundo padre, tal y como se muestra en la Figura 2.10. En la literatura, el término operador de cruce uniforme se relaciona con

Máscara de cruce	1 0 0 1 0 0 1
Padre 1	1 1 0 1 1 0 1
Descendiente	1 0 0 1 1 1 1
Padre 2	0 0 0 1 1 1 0

Figura 2.10: Operador de cruce uniforme

la obtención de la "máscara de cruce" uniforme, en el sentido de que cualquiera de los elementos del alfabeto tenga asociada la misma probabilidad. Hablando en términos de la teoría de la probabilidad la máscara de cruce está compuesta por una muestra aleatoria de tamaño λ extraída de una distribución de probabilidad de Bernouilli de

parámetro 1/2.

Si tuviésemos en cuenta el valor de la función de adaptación de cada padre en el momento de generar la "máscara de cruce", de tal manera que cuanto mayor sea la función de adaptación de un individuo, más probable sea heredar sus características, podríamos definir, véase Larrañaga y Poza (1994), un *operador de cruce basado en la función objetivo*, en el cual la "máscara de cruce" se interpreta como una muestra aleatoria de tamaño l proveniente de una distribución de Bernouilli de parámetro

$$p = g(I_t^j)/(g(I_t^j) + g(I_t^i))$$

donde I_t^j y I_t^i denotan los padres seleccionados para ser cruzados.

El concepto de "máscara de cruce" puede también servir para representar los cruces basados en un punto y basados en múltiples puntos, tal y como se muestra en Figura 2.11

Sirag y Weiser (1987), modifican el *operador de cruce en el sentido del Simulated*

Máscara de cruce	1 1 1 0 0 0 0	1 1 0 0 0 1
Padre 1	1 0 1 1 0 0 1	1 0 1 1 0 0
Descendiente	1 0 1 0 1 1 1	1 0 0 0 1 0
Padre 2	1 0 0 0 1 1 1	1 0 0 0 1 1

Figura 2.11: "Máscaras de cruce" para los operadores de cruce basados en 1 punto y en 2 puntos

Annealing. De esta manera el operador de cruce se modifica definiendo un umbral de energia θ_c, y una temperatura T, las cuales influencian la manera en la que se escogen los bits individuales. Según el operador propuesto el bit $(i+1)$-ésimo se tomará del padre opuesto al que se ha tomado el bit i-ésimo, con probabilidad $\exp(-\theta_c/T)$, donde T es el parámetro "temperatura" el cual, al igual que en *Simulated Annealing* decrecerá lentamente por medio de un programa de enfriamiento. Con altas temperaturas el comportamiento se asemeja al del operador de cruce uniforme, es decir con probabilidad cercana a la unidad los bits se van escogiendo alternativamente de cada padre. Por otra parte cuando el valor del parámetro temperatura se acerca a cero el hijo resultante coincide prácticamente con uno de los padres.

Existen otros operadores de cruce específicos para un determinado problema como son, por ejemplo, los definidos para el TSP, que se tratarán más adelante.

Por otra parte, la idea de que el cruce debería de ser más probable en algunas posiciones ha sido descrita por varios autores (Schaffer y Morishima, 1987; Holland, 1975; Davis, 1985; Levenick, 1991).

2.3.5 Mutación

La mutación se considera un operador básico, que proporciona un pequeño elemento de aleatoridad en la vecindad (entorno) de los individuos de la población. Si bien se admite que el operador de cruce es el responsable de efectuar la búsqueda a lo largo del espacio de posibles soluciones, también parece desprenderse de los experimentos efectuados por varios investigadores que el operador de mutación va ganando en importancia a medida que la población de individuos va convergiendo (Davis, 1985).

Schaffer y col. (1989) encuentran que el efecto del cruce en la búsqueda es inferior

al que previamente se esperaba. Utilizan la denominada *evolución primitiva*, en la cual, el proceso evolutivo consta tan sólo de selección y mutación. Encuentran que dicha evolución primitiva supera con creces a una evolución basada exclusivamente en la selección y el cruce. Otra conclusión de su trabajo es que la determinación del valor óptimo de la probabilidad de mutación es mucho más crucial que el relativo a la probabilidad de cruce.

La búsqueda del valor óptimo para la probabilidad de mutación, es una cuestión que ha sido motivo de varios trabajos. Así, De Jong (1975) recomienda la utilización de una probabilidad de mutación del bit de l^{-1}, siendo l la longitud del string. Schaffer y col. (1989) utilizan resultados experimentales para estimar la tasa óptima proporcional a $1/\lambda^{0,9318}l^{0,4535}$, donde λ denota el número de individuos en la población.

Si bien en la mayoría de las implementaciones de Algoritmos Genéticos se asume que tanto la probabilidad de cruce como la de mutación permanecen constantes, algunos autores han obtenido mejores resultados experimentales modificando la probabilidad de mutación a medida que aumenta el número de iteraciones. Pueden consultarse los trabajos de Ackley (1987), Bramlette (1991), Fogarty (1989) y Michalewicz y Janikow (1991).

2.3.6 Reducción

Una vez obtenidos los individuos descendientes de una determinada población en el tiempo t, el proceso de reducción al tamaño original, consiste en escoger λ individuos de entre los λ individuos que forman parte de la población en el tiempo t, y los λ individuos descendientes de los mismos. Dicho proceso se suele hacer fundamentalmente de dos formas distintas.

O bien los λ individuos descendientes son los que forman parte de la población en el tiempo $t+1$, es lo que se denomina *reducción simple*, o bien se escogen de entre los 2λ individuos, los λ individuos más adaptados al problema, siguiendo lo que podemos denominar un criterio de *reducción elitista de grado* λ. Podemos tambien considerar otros procedimientos de reducción que se colocan entre los anteriores, por ejemplo, si escogemos los λ_1 mejores de entre padres y descendientes, escogiéndose los $\lambda - \lambda_1$ restantes de entre los descendientes no seleccionados hasta el momento.

El concepto de reducción está ligado con el de *tasa de reemplazamiento generacional*, t_{rg}, es decir en el porcentaje de hijos generados con respecto del tamaño de la población.

Si bien en la idea primitiva de Holland (1975) dicho reemplazamiento se efectuaba de 1 en 1, es decir $t_{\mathrm{rg}} = \lambda^{-1}$, habitualmente dicho reemplazamiento se efectúa en bloque, $t_{\mathrm{rg}} = 1$. De Jong (1975) introdujo el concepto de tasa de reemplamiento generacional con el objetivo de efectuar un solapamiento controlado entre padres e hijos. En su trabajo, en cada paso una proporción, t_{rg}, de la población es seleccionada para ser cruzada. Los hijos resultantes podrán reemplazar a miembros de la población anterior. Este tipo de Algoritmos Genéticos se conocen bajo el nombre de *SSGA* (*Steady State Genetic Algorithm*), un ejemplo de los cuales lo constituye *GENITOR* (Whitley y Kauth, 1988; Whitley, 1989).

Michalewicz (1992) introduce un algoritmo que denomina *Algoritmo Genético Modificado*, MOD_{GA}, en el cual para llevar a cabo el reemplazamiento generacional, selecciona al azar r_1 individuos para la reproducción, así como r_2 individuos (distintos

de los anteriores) destinados a morir. Estas selecciones aleatorias tienen en consideración el valor de la función objetivo de cada individuo, de tal manera que cuanto mayor es la función objetivo, mayor es la probabilidad de que sea seleccionado para la reproducción, y menor es la probabilidad de que dicho individuo fallezca. El resto de los $\lambda - (r_1 + r_2)$ individuos son considerados como neutros y pasan directamente a formar parte de la población en la siguiente generación.

2.4 Algoritmos Genéticos Paralelos

En este apartado se introducirán tres maneras diferentes de explotar el paralelismo de los Algoritmos Genéticos, por medio de los denominados modelos de islas. Para una profundización sobre el tema puede consultarse Stender (1993).

Modelos de islas. La idea básica consiste en dividir la población total en varias subpoblaciones en cada una de las cuales se lleva a cabo un Algoritmo Genético. Cada cierto número de generaciones, se efectúa un intercambio de información entre las subpoblaciones, proceso que se denomina *emigración*. La introducción de la emigración hace que los *modelos de islas* sean capaces de explotar las diferencias entre las diversas subpoblaciones, obteniéndose de esta manera una fuente de diversidad genética. Cada subpopulación es una "isla", definiéndose un procedimiento por medio del cual se mueve el material genético de una "isla" a otra. La determinación de la tasa de migración, es un asunto de capital importancia, ya que de ella puede depender la convergencia prematura de la búsqueda.

Se pueden distinguir diferentes *modelos de islas* en función de la comunicación entre las subpoblaciones. Algunas comunicaciones típicas son las siguientes:

- *Comunicación en estrella*, en la cual existe una subpoblación que es seleccionada como *maestra* (aquella que tiene mejor media en el valor de la función objetivo), siendo las demás consideradas como *esclavas*. Todas las subpoblaciones esclavas mandan sus h_1 mejores individuos ($h_1 \geq 1$) a la subpoblación maestra la cual a su vez manda sus h_2 mejores individuos ($h_2 \geq 1$) a cada una de las subpoblaciones esclavas. Véase Figura 2.12.

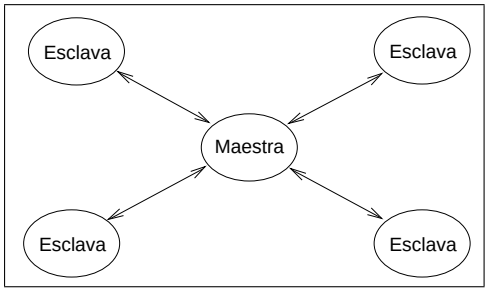

Figura 2.12: Algoritmo Genético Paralelo. Modelo de Islas. Comunicación en estrella

- *Comunicación en red*, en la cual no existe una jerarquía entre las subpoblaciones, mandando todas y cada una de ellas sus h_3 ($h_3 \geq 1$) mejores individuos al resto de las subpoblaciones. Véase Figura 2.13.
- *Comunicación en anillo*, en la cual cada subpoblación envía sus h_4 mejores individuos ($h_4 \geq 1$), a una población vecina, efectuándose la migración en un único sentido de flujo. Véase Figura 2.14.

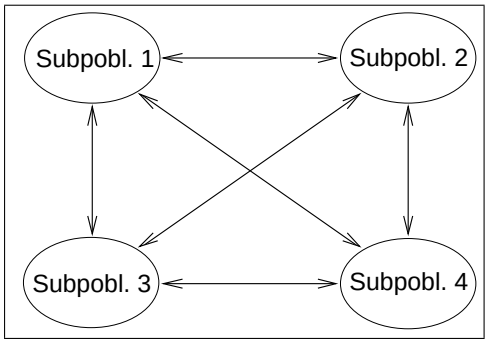

Figura 2.13: Algoritmo Genético Paralelo. Modelo de Islas. Comunicación en red

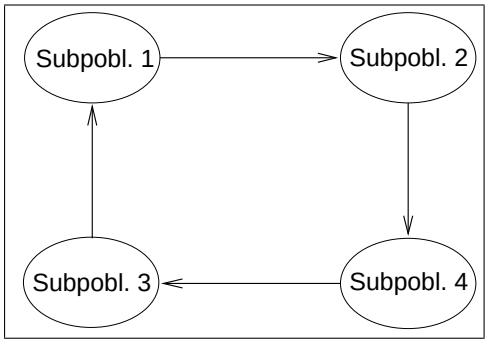

Figura 2.14: Algoritmo Genético Paralelo. Modelo de Islas. Comunicación en anillo

El *modelo de islas* ha sido utilizado por varios autores (Whitley y Starkweather, 1990; Gorges-Schleuter, 1989; Tanese, 1987).

2.5 Evaluación de Algoritmos Genéticos

Las tres medidas de evaluación que se tratarán en este apartado, fueron introducidas por De Jong (1975), y se conocen como:

- evaluación *on-line*
- evaluación *off-line*
- evaluación *basada en el mejor*

Si denotamos por $v_i(t)$ la función objetivo del i-ésimo individuo ($i = 1, \ldots, \lambda$) en la t-ésima población, la evaluación *on-line*, después de T iteraciones, se denotará por $v^{\text{on-line}}(T)$, y se define como

$$v^{\text{on-line}}(T) = \frac{\sum_{t=1}^{T} \sum_{i=1}^{\lambda} v_i(t)}{\lambda T}.$$

Es decir, la evaluación *on-line* mide el comportamiento medio de todas las ristras generadas hasta el tiempo T.

La evaluación *off-line* se refiere al comportamiento del Algoritmo Genético en su proceso de convergencia hacia el óptimo. Así tendremos que si denotamos por $v^*(t)$ al mejor valor de la función objetivo obtenido hasta el tiempo t (incluyendo dicho tiempo), y si $v^{\text{off-line}}(T)$ denota la evaluación *off-line* después de T generaciones, tenemos

que

$$v^{\text{off-line}}(T) = \frac{\sum_{t=1}^{T} v^*(t)}{T}.$$

La definición de evaluación *basada en el mejor* trata de evaluar el Algoritmo Genético por medio del mejor valor de la función de evaluación encontrado en la evolución. Se trata por tanto de la medida de evaluación usual al tratar de estudiar el comportamiento de cualquier técnica heurística.

2.6 Operadores genéticos en el problema del agente viajero

2.6.1 Introducción

El problema del agente viajero, también denominado *TSP* (*Travelling Salesman Problem*), consiste en -dada una colección de ciudades- determinar la gira de mínimo costo, visitando cada ciudad exactamente una vez y volviendo al punto de partida.

Más precisamente, dado un número entero $n \geq 3$ y dada una matriz $C = (c_{ij}) \in M(n, n)$, con elementos c_{ij} enteros no negativos, se trata de encontrar la permutación cíclica π de los enteros de 1 a n que minimiza $\sum_{i=1}^{n} c_{\pi(i)\pi(i+1)}$ con $\pi(n+1) = \pi(1)$.

A lo largo de los años el problema del agente viajero ha ocupado la mente de numerosos investigadores. Los motivos son varios. En primer lugar, el *TSP* es un problema muy sencillo de enunciar, pero muy difícil de resolver. En segundo lugar, el *TSP* es aplicable a una gran variedad de problemas de planificación. Finalmente, se ha convertido en una especie de problema test, es decir los nuevos métodos de optimización combinatoria son a menudo aplicados al *TSP* con objeto de tener una idea de sus potencialidades.

Son numerosos los heurísticos que se han desarrollado para el *TSP*. La mayoría de ellos se describen en Lawler y col. (1985). La primera aproximación al *TSP* a partir de Algoritmos Genéticos la efectuó Brady (1985). Su intento fue seguido por Grefenstette y col. (1985), Goldberg y Lingle (1985), Oliver y col. (1987) y otros muchos. También han sido aplicados otros Algoritmos Evolutivos, véanse por ejemplo, Fogel (1988), Banzhaf (1990) y Ambati y col. (1991).

2.6.2 Representación basada en la trayectoria

La representación basada en la trayectoria es la representación más natural de una gira. En ella, una gira se representa como una lista de n ciudades. Si la ciudad i es el j-ésimo elemento de la lista, la ciudad i es la j-ésima ciudad a visitar. Así por ejemplo, la gira $3-2-4-1-7-5-8-6$ se representará como

$$(3\,2\,4\,1\,7\,5\,8\,6).$$

Debido a que los operadores clásicos no funcionan con esta representación, se han definido otros operadores de cruce y mutación, algunos de los cuales se describen a continuación. Resultados experimentales con dichos operadores de cruce y mutación pueden consultarse en Larrañaga et al. (1999).

2.6.2.A Operadores de cruce

2.6.2.A.1 Operador de cruce basado en una correspondencia parcial (PMX)

El PMX lo introdujeron Goldberg y Lingle (1985). En él, una parte de la ristra representando a uno de los padres, se hace corresponder con una parte, de igual tamaño, de la ristra del otro padre, intercambiándose la información restante.

Por ejemplo, si consideramos los dos padres siguientes:

$$(1\,2\,3\,4\,5\,6\,7\,8) \text{ y}$$

$$(3\,7\,5\,1\,6\,8\,2\,4),$$

el operador PMX crea las giras descendientes de la siguiente manera. En primer lugar, selecciona con probabilidad uniforme dos puntos de corte a lo largo de las ristras que representan las giras padres. Supongamos que el primer punto de corte se selecciona entre el tercer y el cuarto elemento de la gira, y el segundo entre el sexto y el séptimo elemento:

$$(1\,2\,3\,|\,4\,5\,6\,|\,7\,8) \text{ y}$$

$$(3\,7\,5\,|\,1\,6\,8\,|\,2\,4).$$

Se considera que existe una correspondencia biunívoca entre los elementos que forman parte de las subristras comprendidas entre los puntos de corte. En nuestro ejemplo la correspondencia establecida es la siguiente: $4 \leftrightarrow 1$, $5 \leftrightarrow 6$ y $6 \leftrightarrow 8$. A continuación la subristra del primer padre se copia en el segundo hijo. De forma análoga, la subristra del segundo padre se copia en el primer hijo, obteniéndose:

$$\text{descendiente 1: } (x\,x\,x\,|\,1\,6\,8\,|\,x\,x) \text{ y}$$

$$\text{descendiente 2: } (x\,x\,x\,|\,4\,5\,6\,|\,x\,x).$$

En el siguiente paso el descendiente i-ésimo (i=1,2) se rellena copiando los elementos del i-ésimo padre. En el caso de que una ciudad esté ya presente en el descendiente, se reemplaza teniendo en cuenta la correspondencia anterior. Por ejemplo el primer elemento del descendiente 1 será un 1 al igual que el primer elemento del primer padre. Sin embargo, al existir un 1 en el descendiente 1 y teniendo en cuenta la correspondencia $1 \leftrightarrow 4$, se escoge la ciudad 4 como primer elemento del descendiente 1. El segundo, tercer y séptimo elementos del descendiente 1 pueden escogerse del primer padre. Sin embargo, el último elemento del descendiente 1 debería ser un 8, ciudad ya presente. Teniendo en cuenta las correspondencias $8 \leftrightarrow 6$, y $6 \leftrightarrow 5$, se escoge en su lugar un 5. De ahí que

$$\text{descendiente 1: } (4\,2\,3\,|\,1\,6\,8\,|\,7\,5).$$

En forma análoga, se obtiene:

$$\text{descendiente 2: } (3\,7\,8\,|\,4\,5\,6\,|\,2\,1).$$

2.6.2.A.2 Operador de cruce basado en ciclos (CX)

El operador CX (Oliver y col., 1987) crea un descendiente a partir de los padres, de tal manera que cada posición se ocupa por el correspondiente elemento de uno de los padres. Por ejemplo, considerando los padres

$$(1\,2\,3\,4\,5\,6\,7\,8) \text{ y}$$

$$(2\,4\,6\,8\,7\,5\,3\,1),$$

escogemos el primer elemento del descendiente bien del primer elemento del primer padre o del primer elemento del segundo padre. Por tanto, el primer elemento del

descendiente debe ser un 1 o un 2. Supongamos que escogemos el 1. Por el momento, el descendiente tendrá la siguiente forma:

$$(1 * * * * * * *).$$

A continuación debemos de considerar el último elemento del descendiente. Ya que dicho elemento debe ser escogido de uno de los padres, tan sólo puede tratarse de un 8 o un 1. Al haber sido seleccionado el 1 con anteriordad, se escoge el 8, con lo cual el descendiente estará constituido por

$$(1 * * * * * * 8).$$

De forma análoga encontramos que el segundo y cuarto elemento del descendiente deben de ser seleccionados del primer padre, lo cual resulta

$$(1\,2 * 4 * * * 8).$$

Una vez concluido ese ciclo, consideramos a continuación el tercer elemento del descendiente. Dicho elemento puede ser escogido de cualquiera de los padres. Supongamos que lo seleccionamos del segundo padre. Esto implica que los elementos quinto, sexto y séptimo del descendiente deben de escogerse del segundo padre, ya que constituyen un ciclo. De ahí que se obtenga el siguiente descendiente

$$(1\,2\,6\,4\,7\,5\,3\,8).$$

2.6.2.A.3 Operador de cruce basado en el orden (OX1)

El operador OX1 propuesto por Davis (1985), construye descendientes escogiendo una subgira de un padre y preservando el orden relativo de las ciudades del otro padre.

Por ejemplo, considerando las dos giras padres anteriores:

$$(1\,2\,3\,4\,5\,6\,7\,8) \text{ y}$$

$$(2\,4\,6\,8\,7\,5\,3\,1),$$

y suponiendo que se escoge un primer punto de corte entre el segundo y el tercer elemento y un segundo punto entre el quinto y el sexto elemento, se tiene

$$(1\,2\,|\,3\,4\,5\,|\,6\,7\,8) \text{ y}$$

$$(2\,4\,|\,6\,8\,7\,|\,5\,3\,1).$$

Los descendientes se crean de la siguiente manera. En primer lugar, las subgiras comprendidas entre los puntos de corte se copian en los descendientes, obteniéndose

$$(* *\,|\,3\,4\,5\,|\,* * *) \text{ y}$$

$$(* *\,|\,6\,8\,7\,|\,* * *).$$

A continuación, comenzando por el segundo punto de corte de uno de los padres, el resto de las ciudades se copian en el orden en el que aparecen en el otro padre, omitiéndose las ciudades ya presentes. Cuando se llega al final de la ristra de la gira padre, se continúa en su primera posición. En nuestro ejemplo, esto da origen a los siguientes hijos:

$$(8\,7\,|\,3\,4\,5\,|\,1\,2\,6) \text{ y}$$

$$(4\,5\,|\,6\,8\,7\,|\,1\,2\,3).$$

2.6.2.A.4 Operador de cruce basado en el orden (OX2)

Syswerda (1991) sugiere, en conexión con problemas de secuenciación de tareas, el operador OX2, el cual puede considerarse como una modificación del OX1. El operador OX2 selecciona al azar varias posiciones en el string de uno de los padres, para a continuación imponer en el otro padre, el orden de los elementos en las posiciones seleccionadas.

Por ejemplo consideremos los padres

$$(1\,2\,3\,4\,5\,6\,7\,8) \text{ y}$$

$$(2\,4\,6\,8\,7\,5\,3\,1),$$

y supongamos que en el segundo padre se seleccionan las posiciones segunda, tercera y sexta. Los elementos en dichas posiciones son 4, 6 y 5 respectivamente. En el primer padre dichos elementos se encuentran en las posiciones cuarta, quinta y sexta. El descendiente coincidirá con el primer padre si exceptuamos las posiciones cuarta, quinta y sexta:

$$(1\,2\,3\,*\,*\,*\,7\,8).$$

A continuación rellenamos los huecos del descendiente teniendo en cuenta el orden con el que aparecen en el segundo padre. Como resultado obtenemos

$$(1\,2\,3\,4\,6\,5\,7\,8).$$

Cambiando el papel entre el primer y segundo padre, y utilizando las mismas posiciones seleccionadas, obtendremos el segundo descendiente:

$$(2\,4\,3\,8\,7\,5\,6\,1).$$

2.6.2.A.5 Operador de cruce basado en la posición (POS)

Syswerda (1991) propone también en conexión con problemas de secuenciación, una segunda modificación del operador OX1: el operador POS. El operador POS también comienza seleccionando al azar un conjunto de posiciones en las giras padres. Sin embargo este operador impone, la posición de los elementos seleccionados, en los correspondientes elementos del otro padre.

Por ejemplo, si consideramos los padres

$$(1\,2\,3\,4\,5\,6\,7\,8) \text{ y}$$

$$(2\,4\,6\,8\,7\,5\,3\,1),$$

y suponemos que se seleccionan las posiciones segunda, tercera y sexta, esto nos proporcionaría los siguientes descendientes:

$$(1\,4\,6\,2\,3\,5\,7\,8) \text{ y}$$

$$(4\,2\,3\,8\,7\,6\,5\,1).$$

2.6.2.A.6 Operador de cruce basado en la combinación de arcos (ER)

Este operador desarrollado por Whitley y col. (1991), Whitley y Starkweather (1990), utiliza una "conexión de arcos", la cual proporciona para cada ciudad los arcos de los padres que comienzan o finalizan en ella. Por ejemplo, si consideramos las giras:

$$(1\,2\,3\,4\,5\,6) \text{ y}$$

$$(2\,4\,3\,1\,5\,6),$$

la "conexión de arcos" correspondiente puede consultarse en la Tabla 2.3. El operador

ciudad	ciudades conectadas
1	2, 6, 3, 5
2	1, 3, 4, 6
3	2, 4, 1
4	3, 5, 2
5	4, 6, 1
6	1, 5, 2

Tabla 2.3: Conexión de arcos, para las giras $(1\,2\,3\,4\,5\,6)$ y $(2\,4\,3\,1\,5\,6)$

ER funciona de acorde con el siguiente algoritmo:

1. Escoger la ciudad inicial de una de las dos giras padres. Esta selección puede realizarse al azar o de acorde con el criterio expuesto en el paso 4. La ciudad seleccionada se denominará "ciudad de referencia".
2. Quitar todas las ocurrencias de la "ciudad de referencia" de la parte derecha de la tabla de "conexión de arcos" correspondiente.
3. Si la "ciudad de referencia" tiene entradas en la lista de arcos se irá al paso 4, en caso contrario al paso 5.
4. Determinar la ciudad que perteneciendo a la lista de ciudades conectadas con la "ciudad de referencia" tenga el menor número de entradas en su lista de arcos. Dicha ciudad se convierte en la nueva "ciudad de referencia". Los empates se dilucidan al azar. Ir al paso 2.
5. Si no quedan ciudades por visitar, parar el algoritmo. En otro caso, escoger al azar una ciudad no visitada e ir al paso 2.

Para el ejemplo anterior, obtenemos:

- La primera gira descendiente se inicializa con una de las dos ciudades iniciales de sus giras padres. Las ciudades iniciales 1 y 2 ambas tienen 4 arcos; escogemos al azar la ciudad 2.
- La lista de ciudades para la ciudad 2 indica que los candidatos para convertirse en la siguiente "ciudad de referencia" son las ciudades 1, 3, 4 y 6. Las ciudades 3, 4 y 6 tienen todas 2 arcos: los tres iniciales menos la conexión con la ciudad 2. La ciudad 1 tiene tres arcos y por tanto no se tiene en cuenta. Supongamos que se escoge al azar la ciudad 3.

- La ciudad 3 está conectada con la ciudad 1 y con la ciudad 4. Se escoge la ciudad 4 ya que es la que menos arcos tiene.
- La ciudad 4 tan sólo tiene un arco, a la ciudad 5, la cual se escoge a continuación como nueva "ciudad de referencia".
- La ciudad 5 tiene arcos a las ciudades 1 y 6, las cuales tienen ambas tan sólo 1 arco. Escogemos al azar la ciudad 1.
- La ciudad 1 debe ir a la ciudad 6.

La gira resultante es

$$(2\,3\,4\,5\,1\,6),$$

la cual ha sido creada totalmente utilizando arcos tomados de las dos giras padres.

2.6.2.A.7 Operador de cruce basado en la combinación del voto (VR)

El operador VR propuesto por Mühlenbein (1989) puede verse como un operador de cruce p-sexual, donde p es un número natural mayor o igual que 2. Comienza definiendo un umbral, que es un número natural menor o igual que p. A continuación para cada $i \in \{1, 2, \ldots, n\}$ se considera el conjunto de los i-ésimos elementos. Si en dicho conjunto un elemento ocurre con una frecuencia superior o igual al umbral anteriormente determinado, dicho elemento se copia en el descendiente.

Por ejemplo, si se consideran los padres (p=4)

$$(1\,4\,3\,5\,2\,6), (1\,2\,4\,3\,5\,6),$$

$$(3\,2\,1\,5\,4\,6), (1\,2\,3\,4\,5\,6)$$

y definimos el umbral en 3, encontramos

$$(1\,2\,*\,*\,*\,6).$$

El resto de las posiciones se completan con mutaciones. Por tanto, en nuestro ejemplo se podría obtener como resultado:

$$(1\,2\,4\,5\,3\,6).$$

2.6.2.A.8 Operador de cruce basado en la alternancia de las posiciones (AP)

El operador AP (Larrañaga y col., 1999), crea un descendiente seleccionando ciudades alternativamente del primer y segundo padre en el orden ocupado por los mismos, omitiéndose aquellas ciudades que ya se encuentran presenten en la gira descendiente.

Por ejemplo, si el primer padre es

$$(1\,2\,3\,4\,5\,6\,7\,8)$$

y el segundo padre es

$$(3\,7\,5\,1\,6\,8\,2\,4),$$

el operador AP proporciona el siguiente descendiente

$$(1\,3\,2\,7\,5\,4\,6\,8).$$

Cambiando el orden de los padres se obtiene

$$(3\,1\,7\,2\,5\,4\,6\,8).$$

2.6.2.B Operadores de mutación

2.6.2.B.1 Operador de mutación basado en el desplazamiento (DM)

El operador DM (Michalewizc, 1997) comienza seleccionando una subristra al azar. Dicha subristra se extrae de la gira, y se inserta en un lugar aleatorio.

Por ejemplo, si consideramos la gira representada por

$$(1\,2\,3\,4\,5\,6\,7\,8),$$

y suponemos que se selecciona la subristra $(3\,4\,5)$, después de quitar dicha subristra tenemos

$$(1\,2\,6\,7\,8).$$

Supongamos que aleatoriamente seleccionamos la ciudad 7 para insertar a partir de ella la subgira extraida. Esto produciría la gira:

$$(1\,2\,6\,7\,3\,4\,5\,8).$$

2.6.2.B.2 Operador de mutación basado en cambios (EM)

El operador EM (Banzhaf, 1990) selecciona al azar dos ciudades en la gira y las cambia.

Por ejemplo, si consideremos la gira representada por

$$(1\,2\,3\,4\,5\,6\,7\,8),$$

y suponemos que seleccionamos al azar la tercera y la quinta ciudad. El resultado del operador EM sobre la gira anterior sería

$$(1\,2\,5\,4\,3\,6\,7\,8).$$

2.6.2.B.3 Operador de mutación basado en la inserción (ISM)

El operador ISM (Fogel, 1993; Michalewizc, 1992) escoge aleatoriamente una ciudad en la gira, para a continuación extraer dicha ciudad de la gira, e insertarla en un lugar seleccionado al azar.

Por ejemplo, si consideramos de nuevo la gira

$$(1\,2\,3\,4\,5\,6\,7\,8),$$

y suponiendo que se seleccione la ciudad 4, para colocarla a continuación de la ciudad 7, el resultado sería

$$(1\,2\,3\,5\,6\,7\,4\,8).$$

2.6.2.B.4 Operador de mutación basado en la inversión simple (SIM)

El operador SIM (Holland, 1975; Grefenstette y col., 1985) selecciona aleatoriamente dos puntos de corte en la ristra, para a continuación revertir la subristra comprendida entre ambos.

Por ejemplo, si consideramos la gira

$$(1\,2\,3\,4\,5\,6\,7\,8),$$

y suponemos que el primer punto de corte se escoge entre la segunda y tercera ciudad, y el segundo punto de corte se escoge entre la quinta y la sexta ciudad, la gira resultante sería

$$(1\,2\,5\,4\,3\,6\,7\,8).$$

2.6.2.B.5 Operador de mutación basado en la inversión (IVM)

El operador IVM (Fogel, 1988, 1993) es similar al operador DM. Se selecciona al azar una subgira, para a continuación y una vez extraída la misma, insertarla en orden contrario en una posición seleccionada aleatoriamente.

Por ejemplo, si consideramos la gira

$$(1\,2\,3\,4\,5\,6\,7\,8),$$

y se supone que se escoge la subgira $(3\,4\,5)$, para insertarla a continuación de la ciudad 7, obtendríamos

$$(1\,2\,6\,7\,5\,4\,3\,8).$$

2.6.2.B.6 Operador de mutación basado en el cambio (SM)

Este operador de mutación, introducido por Syswerda (1991), selecciona una subgira al azar y a continuación cambia el orden de las ciudades de la misma.

Por ejemplo, considerando la gira

$$(1\,2\,3\,4\,5\,6\,7\,8),$$

y suponiendo que se escoge la subgira $(4\,5\,6\,7)$ podríamos obtener como resultado

$$(1\,2\,3\,5\,6\,7\,4\,8).$$

2.6.3 Representación binaria

En una representación binaria de un TSP con n ciudades, cada ciudad se codifica como una subristra de $\lceil \log_2 n \rceil$ bits, donde $\lceil \log_2 n \rceil$ denota la suma entre la parte entera del logaritmo en base 2 de n y la unidad. Una gira de n ciudades, se representará por medio de una ristra de $n\lceil \log_2 n \rceil$ bits.

Por ejemplo, en un TSP de 6 ciudades, las ciudades se representan por medio de subristras de 3 bits (véase Tabla 2.4). Siguiendo la representación binaria definida

i	ciudad i	i	ciudad i
1	000	4	011
2	001	5	100
3	010	6	101

Tabla 2.4: Representación binaria para un TSP con 6 ciudades

en la tabla, la gira $1-2-3-4-5-6$ se representa por medio de

$$(000\,001\,010\,011\,100\,101).$$

Nótese que existen subristras de 3 bits que no corresponden a ninguna ciudad, por ejemplo 110 y 111.

2.6.3.A El cruce clásico

Para mostrar la problemática que surge al aplicar el operador de cruce basado en un punto, se consideran las dos giras siguientes:

$$(000\,001\,010\,011\,100\,101) \text{ y}$$

$$(101\,100\,011\,010\,001\,000).$$

Suponiendo que el punto de cruce escogido al azar se encuentre entre el noveno y el décimo bit, tendríamos:

$$(000\,001\,010\,|\,011\,100\,101) \text{ y}$$

$$(101\,100\,011\,|\,010\,001\,000).$$

La combinación de las distintas partes da como resultado

$$(000\,001\,010\,010\,001\,000) \text{ y}$$

$$(101\,100\,011\,011\,100\,101).$$

Ninguna de las ristras anteriores representan giras legales.

2.6.3.B La mutación clásica

Al aplicar el operador de mutación, cada bit tiene una probabilidad de ser alterado cercana a cero.

Por ejemplo, si consideramos de nuevo la ristra

$$(000\,001\,010\,011\,100\,101),$$

y suponemos que se mutan el primer y segundo bit, obtenemos como resultado la ristra:

$$(110\,001\,010\,011\,100\,101),$$

la cual no representa una gira.

2.6.4 Otro tipo de representación binaria

Whitley y col. (1989, 1991), sugieren una representación binaria diferente de la anterior, en la cual en primer lugar se define una lista ordenada que contiene todos los posibles arcos no dirigidos. Con la ayuda de esta lista, una gira puede escribirse como un vector binario, con una longitud igual al número de arcos en la lista anterior. El i-ésimo elemento del vector binario valdrá 1, si y sólo si, el i-ésimo arco de la lista ordenada es parte de la gira. En caso contrario su valor será 0.

Por ejemplo, para el TSP de 6 ciudades, la lista ordenada podría definirse a partir de la siguiente lista de arcos no dirigidos:

$$(1,2),(1,3),(1,4),(1,5),(1,6),(2,3),(2,4),(2,5),(2,6),$$

$$(3,4),(3,5),(3,6),(4,5),(4,6),(5,6).$$

Entonces la gira $1-2-3-4-5-6$, se representa por el vector binario

$$100011000100101.$$

2.6.5 Representación matricial

En esta sección se introducen dos aproximaciones al problema usando representaciones matriciales binarias.

Fox y McMahon (1987) representan una gira como una matriz en la cual el elemento (i,j) de la misma vale 1, si y sólo si, en la gira la ciudad i se visita con anterioridad a la j. En caso contrario dicho elemento valdrá 0.

Una manera de definir el cruce es por medio de un operador que construye un descendiente, O, a partir de dos padres, P_1 y P_2, de la manera siguiente. En primer lugar, $\forall i,j \in \{1,2,\ldots,n\}$, siendo n el número de ciudades, se define

$$o_{ij} := \begin{cases} 1 & \text{si } p_{1,ij} = p_{2,ij} = 1, \\ 0 & \text{en otro caso.} \end{cases}$$

A continuación, algunos 1-s que tan sólo aparecen en uno de los padres se añaden al descendiente, para finalmente completarse la matriz de manera que el resultado sea una gira legal.

Por ejemplo, las giras padres $2-3-1-4$ y $2-4-1-3$ que se representan por medio de las matrices:

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \text{ y } \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix},$$

después de la primera fase dan como resultado:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Esta matriz puede completarse de 6 maneras diferentes, ya que la única restricción en la gira descendiente es que comienza en la ciudad 2. Una posible gira descendiente es: $2-1-4-3$, la cual se representa por medio de:

$$\begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Fox y McMahon no definen ningún operador de mutación.

Seniw (1991) define otra representación en la cual el elemento (i, j) de la matriz vale 1, si y sólo si, en la gira la ciudad j se visita inmediatamente después de la ciudad i. Esto implica que una gira legal se representa por medio de una matriz en la cual en cada fila y cada columna existe exactamente un 1. Sin embargo, tal y como se verá en el ejemplo siguiente, una matriz que contiene exactamente un 1 en cada fila y en cada columna no representa necesariamente un gira legal.

Así por ejemplo, si consideramos las matrices

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \text{ y } \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

la primera de ellas representa la gira $2-3-1-4$, mientras que la segunda está representando el conjunto de subgiras $\{1-2\}$, $\{3-4\}$.

Además, los operadores de cruce y mutación definidos por Seniw (1991) no garantizan que el descendiente sea una gira legal, de ahí que necesiten *operadores de reparación* que transformen dichos descendientes en giras legales.

2.6.6 Representación basada en una lista de adyacencia

En la representación basada en una lista de adyacencia, una gira se representa como una lista de n ciudades. La ciudad j está en la posición i-ésima, si y sólo si, la gira va de la ciudad i a la j.

Así por ejemplo, la lista de adyacencia

$$(3\,5\,7\,6\,4\,8\,2\,1)$$

representa la gira

$$1-3-7-2-5-4-6-8.$$

Nótese que toda gira tiene una única representación por medio de una lista de adyacencia. Sin embargo, una lista de adyacencia puede no representar una gira.

Así por ejemplo,

$$(3\,5\,7\,6\,2\,4\,1\,8)$$

representa el siguiente conjunto de subgiras:

$$1-3-7,\ 2-5,\ 4-6 \text{ y } 8.$$

Puede comprobarse que con esta representación el operador de cruce basado en un punto puede dar origen a giras ilegales. Para solventar esta problemática se han

definido otros operadores de cruce -Grefenstette y col. (1985)-.

2.6.7 Representación ordinal

En la representación ordinal -Grefenstette y col. (1985)- una gira se representa como una lista, T, de longitud n, denotando n el número de ciudades. El i-ésimo elemento de la lista T, es un número entero comprendido entre 1 y $n - i + 1$. Existe también una lista ordenada de ciudades, L_0, que sirve como punto de referencia inicial, y que posteriormente se actualiza.

Así por ejemplo, sea la lista ordenada de ciudades:

$$L_0 = (1\,2\,3\,4\,5\,6\,7\,8).$$

Sea la gira $1-5-3-2-8-4-7-6$. Su representación ordinal consiste en la ristra:

$$T = (1\,4\,2\,1\,4\,1\,2\,1)$$

la cual se interpreta de la siguiente manera:

- El primer número de T es el 1. Esto significa que para conseguir la primera ciudad de la gira, se debe de coger el primer elemento de la lista L_0 y a continuación quitarlo de dicha lista. La gira parcial es: 1, y la lista ordenada de ciudades actualizada será:
$$L_1 = (2\,3\,4\,5\,6\,7\,8).$$
- El segundo elemento de T es el 4. Para conseguir la segunda ciudad de la gira, debemos de escoger el cuarto elemento de la lista L_1, que es un 5. A continuación quitamos el 5 de la lista L_1. La gira parcial es: $1 - 5$, y la lista ordenada de ciudades actualizada será:
$$L_2 = (2\,3\,4\,6\,7\,8).$$

Si continuamos de manera análoga hasta que quitemos todos los elementos de la lista ordenada de ciudades, obtenemos la gira $1 - 5 - 3 - 2 - 8 - 4 - 7 - 6$.

La ventaja de la representación ordinal es que el operador de cruce basado en un punto funciona.

Referencias

1. D.H. Ackley (1987). *A Connectionist Machine for Genetic Hillclimbing*, Kluwer Academic Publishers.

2. J.T. Alander (1992). On optimal population size of genetic algorithms. *Proceedings CompEuro 1992, Computer Systems and Software Engineering, 6th Annual European Computer Conference,* 65-70.

3. B.K. Ambati, J. Ambati, M.M. Mokhtar (1991). Heuristic combinatorial optimization by simulated Darwinian evolution: A polynomial time algorithm for the traveling salesman problem, *Biological Cybernetics*, **65**, 31-35.

4. J.E. Baker (1987). Reducing bias and inefficiency in the selection algorithm. *Proceedings of the Second International Conference on Genetic Algorithms and Their Applications*, 14-21.

5. W. Banzhaf (1990). The "molecular" traveling salesman, *Biological Cybernetics*, **64**, 7-14.

6. R.M. Brady (1985). Optimization strategies gleaned from biological evolution, *Nature*, **317**, 804-806.

7. M.F. Bramlette (1991). Initialization, mutation and selection methods in genetic algorithms for function optimization. *Proceedings of the Fourth International Conference on Genetic Algorithms*, 100-107.

8. A. Brindle (1991). *Genetic algorithms for function optimization.* Tesis doctoral, Universidad de Alberta, Canada.

9. U.K. Chakraborty, D.G. Dastidar (1993). Using reliability analysis to estimate the number of generations to convergence in genetic algorithms. *Information Processing Letters*, **46**, 199-209.

10. C. Darwin (1859). *On the Origin of Species by Means of Natural Selection,* Murray, London.

11. L. Davis (1985). Applying adaptive algorithms to epistatic domains, en *Proceedings of the International Joint Conference on Artificial Intelligence*, 162-164.

12. L. Davis (ed.) (1991). *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York.

13. T.E. Davis, J.C. Principe (1993). A Markov chain framework for the simple genetic algorithm. *Evolutionary Computation*, **1(3)**, 269-288.

14. K.A. De Jong (1975). *An analysis of the behaviour of a class of genetic adaptive systems.* Tesis doctoral, University of Michigan.

15. A.E. Eiben, E.H.L. Aarts, K.M. Van Hee (1990). Global convergence of genetic algorithms: An infinite Markov chain analysis. *Computing Science Notes*, Eindhoven University of Technology, The Netherlands.

16. T.C. Fogarty (1989). Varying the probability of mutation in the genetic algorithm. *Proceedings of the Third International Conference on Genetic Algorithms,* 104-109.

17. D. B. Fogel (1988). An evolutionary approach to the traveling salesman problem, *Biological Cybernetics*, **60**, 139-144.

18. D. B. Fogel (1993). Applying evolutionary programming to selected traveling salesman problems, *Cybernetics and Systems*, **24**, 27-36.

19. M. S. Fox, M. B. McMahon (1987). Genetic operators for sequencing problems, en Rawlings, G. (ed.) *Foundations of Genetic Algorithms: First Workshop on the Foundations of Genetic Algorithms and Classifier Systems*, Morgan Kaufmann Publishers, Los Altos, CA, 284-300.

20. D.E. Goldberg (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA.

21. D. E. Goldberg, Jr. R. Lingle (1985). Alleles, loci and the traveling salesman problem, en *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, 154-159.

22. D.E. Goldberg, J.T. Richardson (1987). Genetic algorithms with sharing for multimodal function optimization. *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms and Their Applications*, 41-49.

23. M. Gorges-Schleuter (1989). ASPARAGOS An asynchronous parallel genetic optimization strategy. *Proceedings on the Third International Conference on Genetic Algorithms*, 422-427.

24. J. Grefenstette, R. Gopal, B. Rosmaita, D. Van Gucht (1985). Genetic algorithms for the traveling salesman problem, en *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, 160-165.

25. J. Holland (1975). *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor.

26. P. Larrañaga, M. Poza (1994). Structure learning of Bayesian network by genetic algorithms. E. Diday (ed.), *New Approaches in Classification and Data Analysis*, Springer-Verlag, 300-307.

27. P. Larrañaga, C. Kuijpers, R. Murga, I. Inza, S. Dizdarevich (1999) Evolutionary algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, **13**, 129–170.

28. E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys (eds.) (1985). *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley, Chichester.

29. G.E. Liepins (1992). On global convergence of genetic algorithms. *Neural and Stochastic Methods in Image and Signal Processing*, 61-65.

30. Z. Michalewicz (1992). *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, Berlin Heidelberg.

31. Z. Michalewicz, C.Z. Janikow (1991). Handling constraints in genetic algorithms. *Proceedings of the Fourth International Conference on Genetic Algorithms*, 151-157.

32. H. Mühlenbein (1989). Parallel genetic algorithms, population genetics and combinatorial optimization, en *Proceedings on the Third International Conference on Genetic Algorithms*, 416-421.

33. I.M. Oliver, D.J. Smith, J.R.C. Holland (1987). A study of permutation crossover operators on the TSP, en *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference*, 224-230.

34. C. Reeves (1993). *Modern Heuristic Techniques for Combinatorial Problems,* Blackwell Scientific Publications.

35. G. Reinelt (1991). TSPLIB - A Traveling Salesman Library, *ORSA Journal on Computing*, **3(4)**, 376-384.

36. G. Rudolph (1994). Convergence analysis of canonical genetic algoritms. *IEEE Transactions on Neural Networks*, 96-101.

37. J.D. Schaffer, R.A. Caruna, L.J. Eshelman, R. Das (1989). A study of control parameters affecting online performance of genetic algorithms for function optimization. J.D. Schaffer (ed.), *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, 51-60.

38. D. Seniw (1991). A genetic algorithm for the traveling salesman problem, *M.Sc. Thesis*, University of North Carolina at Charlotte.

39. D.J. Sirag, P.T. Weisser (1987). Toward a unified thermodynamic genetic operator. *Genetic Algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms and Their Applications*, 116-122.

40. J. Stender (1993). *Parallel Genetic Algorithms: Theory and Applications,* IOS Press.

41. J. Suzuki (1993). A construction of Bayesian networks from databases based on an MDL Principle. *Uncertainty in Artificial Intelligence, Proceedings of the Ninth Conference*, 266-273.

42. G. Syswerda (1991). Schedule optimization using genetic algorithms. L. Davis (ed.). *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 332-349.

43. R. Tanese (1987). Parallel genetic algorithm for a hypercube. *Proceedings of the Second International Conference on Genetic Algorithms*, 177-183.

44. D. Whitley (1989). The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. *Proceedings on the Third International Conference on Genetic Algorithms*, 116-121.

45. D. Whitley, J. Kauth (1988). GENITOR: A different genetic algorithm. *Proceedings of the Rocky Mountain Conference on Artificial Intelligence*, Denver, CO, 118-130.

46. D. Whitley, T. Starkweather, D. Fuquay (1989). Scheduling problems and travelling salesman: The genetic edge recombination operator, en *Proceedings on the Third International Conference on Genetic Algorithms*, 133-140.

47. D. Whitley, T. Starkweather (1990). Genitor II: A distributed genetic algorithm. *Journal of Experimental and Theorical Artificial Intelligence*, **2**, 189-214.

48. D. Whitley, T. Starkweather, D. Shaner (1991). The traveling salesman and sequence scheduling: Quality solutions using genetic edge recombination, en Davis, L. (ed.) *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 350-372.