CAPÍTULO 6: ALGORITMOS EVOLUTIVOS AVANZADOS PARA TSP

6.1. INTRODUCCIÓN

Los primeros avances para solucionar el TSP, por medio de Algoritmos Evolutivos han sido introducidos por Goldberg y Lingle en [68] y Grefenstette en [72]. En éste área muchos son los esfuerzos realizados para alcanzar las siguientes características [63]:

- Una apropiada representación para codificar un tour.
- Operadores genéticos válidos para mantener los subtours (o subcircuitos) evitando ilegalidad.
- Prevención de la convergencia prematura.

En función a lo anterior se han creado distintos tipos de representaciones, operadores de crossover y mutación. Entre las formas de representación pueden distinguirse las siguientes: *por permutación*, *por claves aleatorias*, *por adyacencia*, *ordinal* [107, 29, 12]. Mientras que los operadores de crossover usados en el TSP son: *PMX*, *OX*, *Crosover basado en la posición*, *Crossover basado en el orden*, *CX*, *Crossover de intercambio de subtuors*, *Crossover heurístico*, [68, 26, 111, 149, 150, 72]. Por último los operadores de mutación utilizados en este problema son: *por inversión*, *por inserción*, *por desplazamiento*, *intercambio recíproco*, *Mutación heurística* [21, 22].

Cabe destacar que el problema de los algoritmos evolutivos basados en operadores de crossover es el siguiente: que requieren mayor tiempo computacional, por lo que resultan algoritmos costosos. Mientras que los algoritmos basados en operadores de mutación no escapan en forma eficiente de los óptimos locales.

En las secciones siguientes se describen cada una de las formas de: representación, crossover y mutación.

6.2. REPRESENTACIÓN DEL CROMOSOMA

Tradicionalmente los cromosomas son cadenas binarias. Esta simple representación no es conveniente para el TSP y otros problemas combinatoriales. Varios esquemas de representación se han propuesto para el TSP; éstos son descriptos en las secciones 6.2.1 a 6.2.4.

6.2.1. REPRESENTACIÓN POR ADYACENCIA

En la representación por adyacencia [107] un tour se conforma como una lista de *n* ciudades. La ciudad *j* está en la posición *i* de la lista si y solo sí el tour se dirige de la ciudad *i* a la ciudad *j*. Por ejemplo, el vector

(2 4 8 3 9 7 1 5 6)

representa el siguiente tour:

1 – 2 – 4 – 3 – 8 – 5 – 9 – 6 – 7

Cada tour sólo tiene una lista de representación por adyacencia, sin embargo algunas listas pueden representar tours ilegales. Por ejemplo, el vector

(2 4 8 1 9 3 5 7 6),

conduce a:

1 – 2 – 4 –1,

un tour parcial con un ciclo prematuro.

La representación por adyacencia no soporta el operador de crossover clásico (crossover de un punto), por lo tanto resulta necesario un algoritmo de reparación. Los operadores de crossover válidos para esta representación son el *crossover de: arcos alternativos*, *subtours chunks* y el heurístico.

6.2.2. REPRESENTACIÓN ORDINAL

La representación ordinal [107] constituye un tour como una lista de *n* ciudades; el *i*-ésimo elemento de la lista es un número en el rango de 1 a $n - i + 1$. La idea detrás de la representación ordinal es como sigue. Dada una lista de ciudades *C*, que sirve como punto de referencia para listas en representaciones ordinales, se asume por ejemplo que tal lista ordenada es canónica:

C = (1 2 3 4 5 6 7 8 9).

Dado un tour

1 – 2 – 4 – 3 – 8 – 5 – 9 – 6 – 7

se representa como una lista *l* de referencias,

l = (1 1 2 1 4 1 3 1 1),

y debería interpretarse como sigue:

- El primer número sobre la lista *l* es 1, entonces toma la primer ciudad desde la lista *c* como la primer ciudad del tour (ciudad número 1), y la elimina desde *C*. El tour parcial es:

  1 –

- El número próximo sobre la lista *l* es también 1, entonces toma la primer ciudad de lista *C* actualizada con la próxima ciudad del tour (ciudad número dos) y la elimina de *C*. El tour parcial es:

  1 – 2

- El número próximo sobre la lista *l* es 2, entonces toma la segunda ciudad de lista *C* actualizada con la próxima ciudad del tour (ciudad número cuatro) y la elimina de *C*. El tour parcial es:

  1 – 2 – 4

- Así siguiendo con cada uno de los elementos de la lista *l* hasta llegar al final de la misma.

La principal ventaja de la representación ordinal es que el operador de crossover clásico (crossover de un punto) funciona. Un ejemplo de esto se muestra en la figura 6.1.

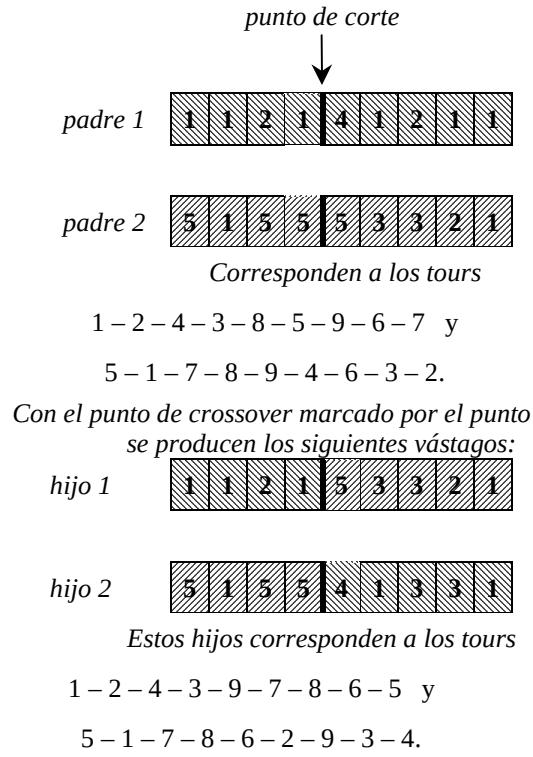

Figura 6.1. Ilustración del operador de crossover clásico bajo una representación ordinal.

Es fácil ver que los tours parciales a la izquierda del punto de crossover no cambian, mientras que los tours parciales a la derecha del punto de crossover son desorganizados de forma bastante aleatoria. Los pobres resultados experimentales

obtenidos [72] indican que esta representación junto con el operador de crossover clásico no es apropiada para el TSP.

6.2.3. REPRESENTACIÓN DEL CAMINO (POR PERMUTACIÓN)

Esta representación es por supuesto la forma más natural de presentar un tour para el TSP, donde las ciudades son listadas en el orden en el cual son visitadas [29, 107]. El espacio de búsqueda para ésta representación es el conjunto de permutaciones de las ciudades. Por ejemplo, un tour para el TSP de 9 ciudades

3 – 2 – 5 – 4 – 7 – 1 – 6 – 9 – 8

es simplemente representado como sigue:

(3 2 5 4 7 1 6 9 8).

Esta representación puede conducir a tours ilegales si se utiliza el crossover de un punto. Muchos operadores de crossover se han investigado por esto. Los más conocidos son: PMX, CX y OX.

6.2.4. REPRESENTACIÓN POR CLAVES ALEATORIAS.

Es la primer representación introducida por Bean [12]. La cual codifica la solución con números aleatorios en el intervalo abierto (0,1). Estos valores se usan como claves de ordenamiento para codificar la solución. Por ejemplo, un cromosoma para un problema de 9 ciudades puede representarse como:

(0.23 0.82 0.45 0.74 0.87 0.11 0.56 0.69 0.78),

donde la posición *i* en la lista representa la ciudad *i*. El número aleatorio en la posición *i* determina el orden de visita de la ciudad *i* en un tour TSP. Para este

ejemplo se ordenan las claves en orden ascendente, obteniéndose así el siguiente tour:

6 – 1 – 3 – 7 – 8 – 4 –9 – 2 – 5

Las claves aleatorias eliminan la posibilidad de hijos no factibles al representar soluciones de la forma antes descripta. Esta representación es aplicable a una gran variedad de problemas de optimización secuenciales incluyendo el scheduling de máquinas, alocación de recursos, ruteo vehicular, problemas de asignación cuadrática.

6.3. OPERADORES DE CROSSOVER

Varios operadores de crossover se han propuesto para el TSP, tales como *Partial-Mapped Crossover* (PMX), *Order Crossover* (OX), *Cycle Crossover* (CX), Crossover Basado en la Posición, Crossover Basado en el Orden, Crossover Heurístico, Crossover de Subtours Chunks, Crossover por Arcos Alternativos (Edge Recombination, *ER*), entre otros. Estos operadores pueden clasificarse en:

- canónicos y
- heurísticos.

Los operadores canónicos es posible verlos como una extensión del crossover de dos puntos al crossover multipunto de tours binarios. Generalmente la representación por permutación produce vástagos ilegales al usar crossover de dos puntos o multipunto, ya que algunas ciudades pueden perderse, mientras que otras pueden duplicarse en el vástago. En estos casos los procedimientos de reparación se usan para resolver la ilegitimidad del hijo.

La esencia de la opción canónica es el mecanismo aleatorio. No hay garantía de que un hijo producido por este tipo de crossover sea mejor que sus padres. La aplicación de heurísticas en el crossover intenta generar un vástago mejorado.

6.3.1. PARTIAL MAPPED-CROSSOVER (PMX)

Goldberg y Lingle presentan PMX en [68]. Es posible ver a este operador como: una extensión del crossover de dos puntos para un tour binario a una representación por permutación. Este usa un procedimiento de reparación especial para resolver la ilegitimidad causada por el simple crossover de dos puntos. Es decir, las bases de PMX son el crossover de dos puntos más un procedimiento de reparación. PMX opera de la siguiente manera:

1. Selecciona, uniformemente al azar, dos posiciones a lo largo del tour. Los *subtours* definidos por las dos posiciones se *denominan secciones de transferencia* (o mapping sections en inglés).
2. Intercambia dos subtours entre los padres para producir 2 hijos.
3. Determina las relaciones de transferencia entre las dos secciones transferidas.
4. Legaliza los hijos con las *relaciones de transferenica*.

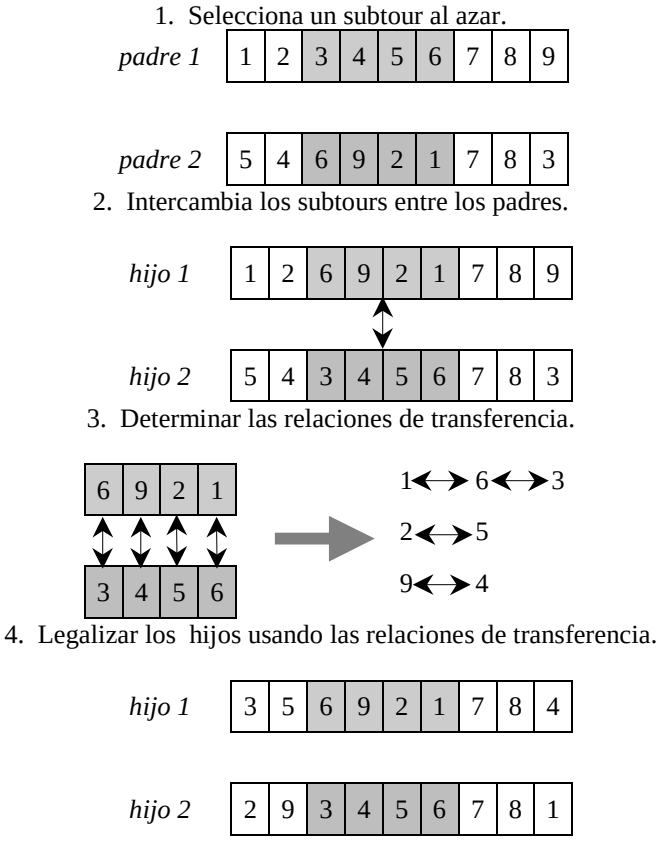

Figura 6.2. Ilustración del operador PMX

La figura 6.2 ilustra el procedimiento del PMX. Las ciudades 1, 2, y 9 están duplicadas en el hijo 1, mientras que las ciudades 3, 4, y 5 son perdidas. De acuerdo a las relaciones de transferencia determinadas en el paso 3, las ciudades repetidas 1,2, y 9 deberían reemplazarse por las ciudades faltantes 3, 5, y 4, respectivamente, mientras mantiene el subtour intercambiado sin modificaciones.

6.3.2. ORDER CROSSOVER (OX)

OX es propuesto por Davis [26]. Este puede verse como una variación de PMX con un procedimiento de reparación diferente. OX trabaja como sigue:

1. Selecciona un subtour desde un padre al azar.
2. Produce un hijo al copiar el subtour dentro sus correspondientes ubicaciones.
3. Borra las ciudades que están en el subtour del segundo padre. La secuencia de ciudades resultantes son las requeridas por los hijos.
4. Ubica las ciudades en las posiciones libres del hijo de izquierda a derecha, de acuerdo al orden de la secuencia para producir un vástago.

La figura 6.3 ilustra este procedimiento. Esta muestra un ejemplo de cómo se obtiene un cromosoma hijo. Con idénticos pasos, podemos producir el segundo hijo [2 5 4 9 1 3 6 7 8] desde los mismos padres.

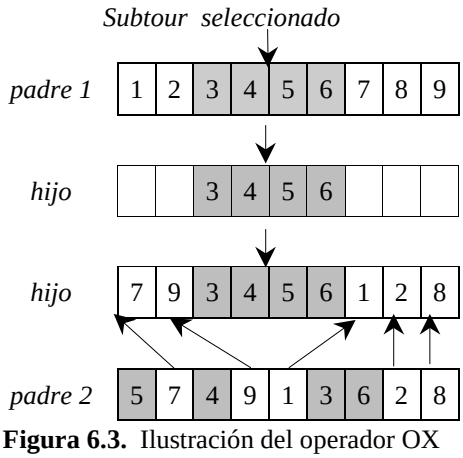

Figura 6.3. Ilustración del operador OX

6.3.3. CROSSOVER BASADO EN LA POSICIÓN

Syswerda presenta el crossover basado en la posición en [26]. Este es esencialmente un tipo de crossover uniforme para una representación por permutación junto con un procedimiento de reparación. Esto puede visualizarse también como una variación de OX en el cual las ciudades se seleccionan en forma no consecutiva. El operador de crossover basado en la posición funciona como sigue:

1. Selecciona un conjunto de posiciones al azar desde un padre.
2. Produce un hijo al copiar las ciudades sobre estas posiciones en las correspondientes ubicaciones del hijo.
3. Borra las ciudades que son seleccionadas desde el segundo padre. La secuencia resultante contiene las ciudades que necesita el hijo.
4. Ubica las ciudades en las posiciones libres del hijo de izquierda a derecha de acuerdo al orden de la secuencia; y así producir un hijo.

El procedimiento anterior se ilustra en la figura 6.4. Con los mismos pasos, se puede producir el segundo hijo como [2 4 3 5 1 9 6 7 8].

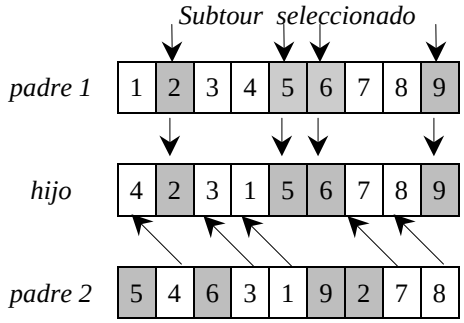

Figura 6.4. Ilustración del operador de crossover basado en la posición.

6.3.4. CROSSOVER BASADO EN EL ORDEN

El operador de crossover basado en el orden lo introduce Syswerda en [26]. Este presenta una leve variación al operador de crossover basado en la posición en la cual el orden de las ciudades seleccionadas en un padre, es impuesta sobre las ciudades correspondientes en el otro padre. Como muestra la figura 6.5. Con los mismos pasos, se puede generar el segundo hijo como [4 2 3 1 5 6 7 9 8].

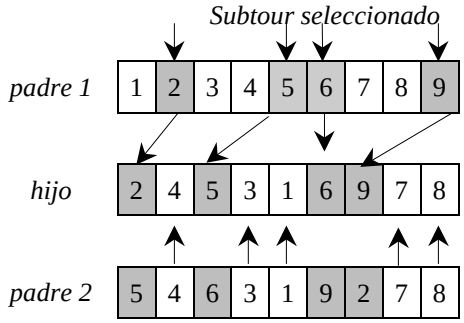

Figura 6.5. Ilustración del operador de crossover basado en la posición.

6.3.5. CYCLE CROSSOVER (CX)

Oliver, Smith y Holland proponen CX en [111]. Al igual que el crossover basado en la posición, este toma las mismas ciudades desde un padre y selecciona las ciudades restantes desde el otro padre. La diferencia radica en que las ciudades del primero no se seleccionan aleatoriamente, sino que se eligen si definen un ciclo de acuerdo a las posiciones correspondientes entre padres. CX funciona de la siguiente [60]forma:

1. Encuentra el ciclo definido por las posiciones correspondientes de las ciudades entre los padres.
2. Copia las ciudades en el ciclo a un hijo con las posiciones correspondientes de un padre.
3. Determina las ciudades restantes para el hijo al eliminar las ciudades que ya están en el ciclo desde el otro padre.
4. Completa al hijo con las ciudades restantes.

Este procedimiento se ilustra en la figura 6.6. Con los mismos pasos se puede crear el segundo vástago.

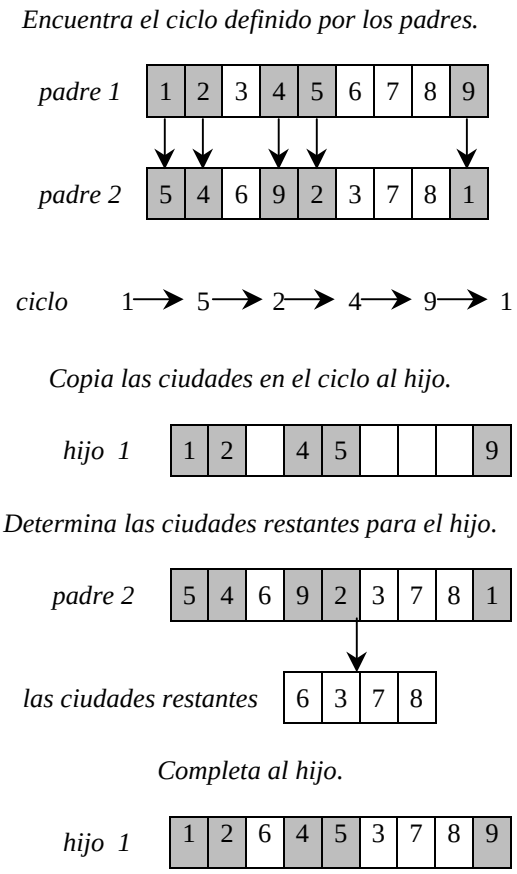

Figura 6.6. Ilustración del operador CX.

6.3.6. SUBTOUR EXCHANGE CROSSOVER

Yamamura, Ono, y Kobayashi proponen este operador de crossover [149, 150]. Primero selecciona subtours desde los padres. Los cuales contienen las ciudades en común. Los hijos son creados al intercambiar los subtours como lo muestra la figura 6.7.

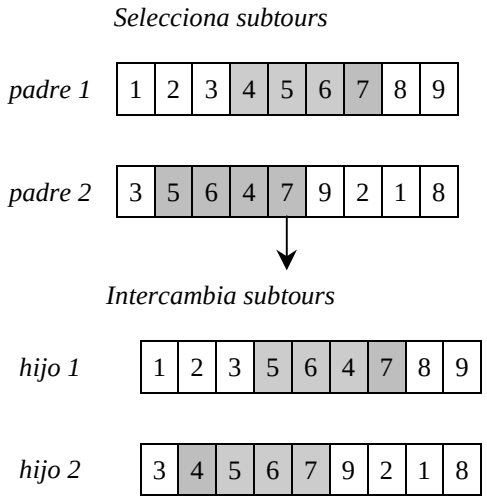

Figura 6.7. Ilustración del operador subtour exchange crossover.

6.3.7. CROSSOVER HEURÍSTICO

El crossover heurístico es propuesto por Grefenstette et. al en [72]. En heurísticas convencionales para el TSP, existen dos alternativas de construcción básicas: heurísticas del *vecino más cercano*, y la de *mejor inserción*. El crossover presentado por Grefenstette se implementa con la heurística del vecino más cercano. Cheng y Gen han diseñado un crossover con el mecanismo de la mejor inserción para el problema del ruteo vehicular y del viajante. El crossover heurístico funciona así:

1. Dado un par de padres se elige una ciudad al azar, para el inicio.
2. Se elige el arco más corto (representado en los padres) que va desde la ciudad actual a aquellas ciudades que no forman un ciclo. Si los dos arcos conducen a un ciclo, se elige, en forma aleatoria, una ciudad que continúe el tour.
3. Si se completa el tour, el procedimiento se detiene; de otra forma se repite el paso 2.

Liepins presenta una leve modificación de la versión de Grefenstette [98], la cual toma cualquier tour siempre que comience en la misma ciudad. Cheng y Gen también han propuesto una versión modificada del crossover heurístico [21]. Esta

presenta un *procedimiento de rotación de subtour* para heredar a los mejores subtours de los padres en los hijos.

6.3.8. CROSSOVER POR ARCOS ALTERNATIVOS

Michalewicz implementa este operador [107] para la representación por adyacencia. Construye un hijo al elegir aleatoriamente un arco desde el primer padre, entonces selecciona el arco apropiado desde el segundo progenitor, etc. El operador extiende el tour al elegir arcos desde los padres alternativos. Si el nuevo arco (desde uno de los padres) introduce un ciclo al tour actual, entonces el operador selecciona uno al azar, que no introduzca un ciclo, desde los arcos restantes.

Por ejemplo, el primer hijo de dos padres puede ser el presentado (*hijo 1*)en la figura 6.8.

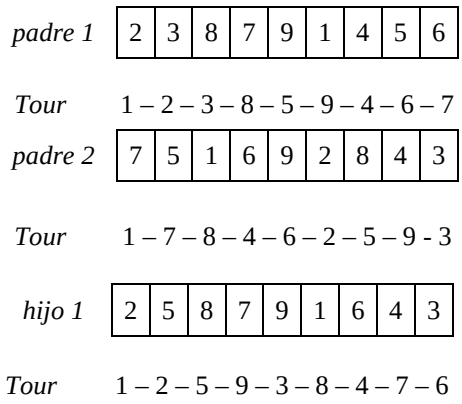

Figura 6.8. Ilustración del operador por arcos alternativos

Donde el proceso se inicia en el arco (1,2) desde el *padre 1* y el único arco introducido aleatoriamente durante el proceso de los arcos alternativos es (7,6) en vez de (7,8). El cual introduciría un ciclo prematuro.

6.3.9. CROSSOVER POR SUBTOURS CHUNKS

El crossover por Subtours Chunks [107], al igual que el presentado en la sección anterior se utiliza con la representación por adyacencia.

Este operador de crossover construye un hijo al elegir un subtour (de longitud aleatoria) desde uno de los padres, y luego se elige otro desde uno de los padres restantes (también de longitud aleatoria). El operador extiende el tour al seleccionar arcos desde padres alternativos. Si algún arco (pertenecientes a alguno de los padres) introduce un ciclo en el tour actual, el operador selecciona, en forma aleatoria, un arco a partir de los arcos restantes que no incluyan ciclos.

6.3.10. CROSSOVER POR RECOMBINACIÓN DE ARCOS

Whitley, Starweather y Fuquay introducen este operador de crossover [146, 107]. ER transfiere más del 95% de los arcos desde los padres a un único hijo y explora la información sobre los arcos en un tour, por ejemplo para el tour

3 – 1 – 2 – 8 – 7 – 4 – 6 – 9 – 5

los arcos son:

(3 1), (1 2), (2 8), (8 7), (7 4), (4 6), (6 9), (9 5) y (5 3).

Luego todos los arcos -no las ciudades- poseen valores (distancias) en el TSP. Entonces la función objetivo, a minimizarse, es el total de arcos que constituyen un tour legal. La posición de una ciudad en un tour no es importante, ya que los tours son circulares. Tampoco lo es la dirección de un arco; porque los arcos (3 1) y (1 3), por ejemplo, indican sólo que las ciudades 1 y 3 están directamente conectadas entre sí.

La idea general detrás del crossover ER es que un hijo debería construirse exclusivamente desde los arcos presentes. Esto se realiza con ayuda de la lista de arcos creada desde los dos tours padres. La lista de arcos provee, por cada ciudad *c*, todas las otras ciudades conectadas a la ciudad *c* en al menos uno de los padres. En todo grafo, por cada ciudad *c* existe al menos dos y a lo sumo 4 ciudades en la lista. Un ejemplo de esto, se observa en la figura 6.9.

padre 1	1	2	3	4	5	6	7	8	9
padre 2	4	1	2	8	7	6	9	3	5

La lista de arcos es:

Ciudad 1: arcos a otras ciudades: 9 2 4
Ciudad 2: arcos a otras ciudades: 1 3 8
Ciudad 3: arcos a otras ciudades: 2 4 9 5
Ciudad 4: arcos a otras ciudades: 3 5 1
Ciudad 5: arcos a otras ciudades: 4 6 3
Ciudad 6: arcos a otras ciudades: 5 7 9
Ciudad 7: arcos a otras ciudades: 6 8
Ciudad 8: arcos a otras ciudades: 7 9 2
Ciudad 9: arcos a otras ciudades: 8 1 6 3

Figura 6.9. Ilustración del operador de crossover ER

La construcción del hijo comienza con la selección de una ciudad de inicio desde uno de los padres. Entonces, se elige la ciudad conectada a la inicial con el menor número de arcos. Si estos números son iguales, se realiza una elección aleatoria entre ellos. Cada selección incrementa la posibilidad de completar un tour con todos los arcos seleccionados desde los padres. Con una selección al azar, la chance de tener un arco erróneo debiera ser mucho más grande.

Se asume que la ciudad 1 es seleccionada. Esta ciudad está directamente conectada con otras tres: 9, 2, y 4. La próxima ciudad se elige desde una de estas tres. En este ejemplo, las ciudades 4 y 2 tienen tres arcos, y la ciudad 9 tiene 4. Se hace una selección aleatoria entre las ciudades 4 y 2; se asume que se ha seleccionado la ciudad 4. Siendo el tour parcial resultante:

1 – 4

Nuevamente los candidatos para la próxima ciudad del tour, que se está construyendo, son 3 y 5, si ellos están directamente conectados a la última ciudad (4), entonces se selecciona la 5. Siendo el tour parcial resultante:

1 – 4 – 5

Continuando con este procedimiento se obtiene el hijo:

1 – 4 – 5 – 6 – 7 – 8 – 2 – 3 – 9

el cual se forma completamente por arcos tomados desde ambos padres.

6.4. OPERADORES DE MUTACIÓN

Es relativamente fácil producir algunos operadores de mutación para representación del camino. Algunos de ellos pueden ser la mutación por: inversión, inserción, desplazamiento e intercambio recíproco [21, 22].

6.4.1. MUTACIÓN POR INVERSIÓN

La mutación por inversión selecciona dos posiciones dentro de un cromosoma al azar y entonces invierte el subtour entre estas dos posiciones, como se ilustra en la figura 6.10.

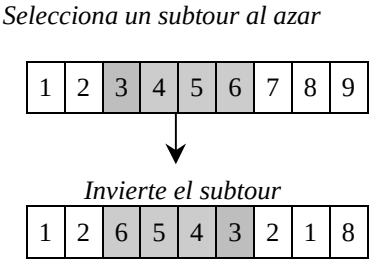

Figura 6.10. Ilustración del operador de mutación por inversión.

6.4.2. MUTACIÓN POR INSERCIÓN

La mutación por inserción selecciona una ciudad al azar y la inserta en una posición aleatoria, como ilustra la figura 6.11.

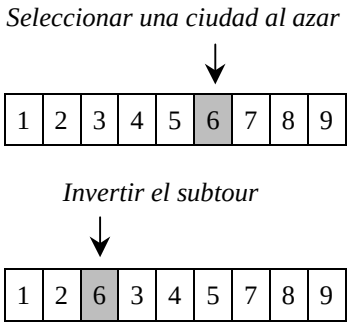

Figura 6.11. Ilustración del operador de mutación por inserción.

6.4.3. MUTACIÓN POR DESPLAZAMIENTO

Este tipo de mutación selecciona un subtour al azar y lo inserta en una posición aleatoria, como se ilustra en la figura 6.12. La inserción puede verse como un caso especial de desplazamiento en el cual el subtour contiene sólo una ciudad.

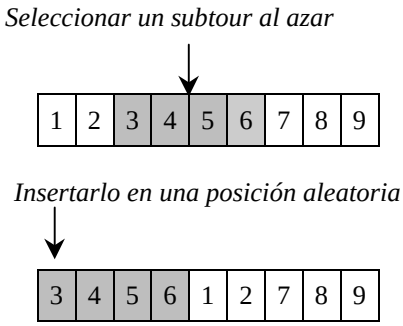

Figura 6.12. Ilustración del operador de mutación por desplazamiento.

6.4.4. MUTACIÓN POR INTERCAMBIO RECÍPROCO

La mutación por intercambio recíproco selecciona dos posiciones aleatorias y luego intercambia las ciudades sobre estas posiciones, como ilustra la figura 6.13. Esta mutación es esencialmente la heurística 2-Opt para TSP.

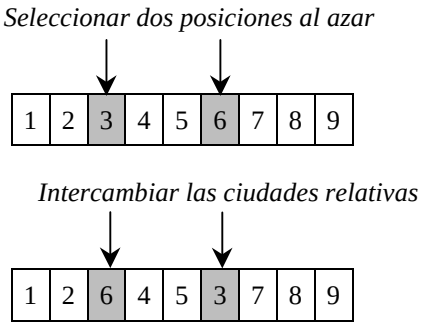

Figura 6.13. Ilustración del operador de mutación por intercambio recíproco.

6.4.5. MUTACIÓN HEURÍSTICA

Cheng y Gen proponen este tipo de mutación [21, 22]. Está diseñada con la técnica del vecindario para producir, así, un hijo mejorado. Un conjunto de cromosomas se transforman a partir de un cromosoma dado al intercambiar no más de λ genes. Este conjunto de cromosomas obtenido es considerado como el vecindario. Entonces, el mejor de estos vecinos es tomado como el hijo producido por la mutación. Este operador sigue los siguientes pasos:

1. Toma λ genes aleatoriamente.
2. Genera vecinos de acuerdo a todas las permutaciones posibles de los genes seleccionados.
3. Evalúa a todos los vecinos y selecciona el mejor como hijo.

La figura 6.14 muestra este procedimiento.

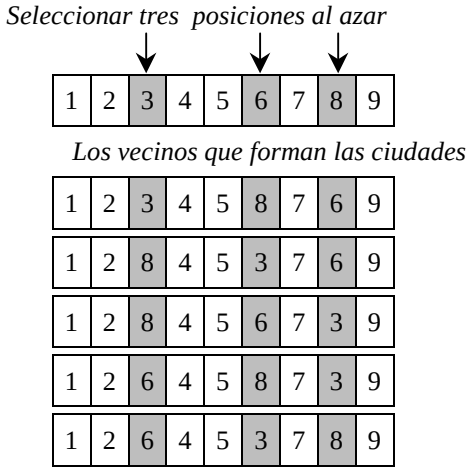

Figura 6.14. Ilustración del operador de mutación heurística