4. Diseño e implementación de métodos de resolución al problema de los SFRC.

En el capítulo anterior hemos descrito el problema al que nos enfrentamos de una forma más concreta y la analogía que existe con un problema bastante tratado en la literatura, el m-TSP. Tras haber mencionado distintos enfoques para resolverlo, en este capítulo nos centraremos en mostrar cómo éstos pueden ser aplicados a nuestro problema concreto, el de los SFRC.

Se desean crear células de fabricación a las que se asignen una o varias familias de productos a fabricar, considerando que dentro de cada célula existirán, por una parte unos costes de reconfiguración al pasar de fabricar una familia a la siguiente, y por otra parte unos costes de no-uso de los recursos dentro de cada familia. El objetivo es decidir las familias de productos que irán en cada célula y el orden de fabricación dentro de las mismas, con mínimo coste total.

El punto de partida del estudio a realizar será el dendograma obtenido en nuestra investigación previa. En base al mismo, se desea formar un número de células de fabricación conocido, cada uno de ellas con una o más familias asignadas. Para cada célula, se obtendrá una secuencia con el orden de fabricación de las familias, incurriendo en unos costes de reconfiguración y en costes de no-uso de los recursos dentro de cada familia de las células.

Si el número de células a formar es uno, el problema coincidirá con el ya estudiado de SFR, es decir, la solución final seleccionará un nivel del dendograma y ordenará las familias del mismo a mínimo coste. En esta investigación se modificará el modelo anterior en el siguiente aspecto: se establecerá un estado inicial de cada célula al que hay que volver tras la fabricación de todas las familias asignadas a la célula, en lugar de volver a la familia inicial. Consideraremos que este estado inicial no consume recursos del sistema, por lo que puede verse como una familia virtual. Esto fue introducido e ilustrado en la Figura 28 y será justificado en el apartado 4.1.2.

Si se asignase una única familia a cada célula el problema se reduce a seleccionar el nivel del dendograma que coincida con el número de células a formar, no siendo necesario el modelo de secuenciación de familias por célula. Dicha solución da lugar a mínimo coste de reconfiguración pero máximo coste de no-uso de recursos. Las investigaciones realizadas hasta el momento no

consideran directamente los costes de reconfiguración, dando lugar a una única familia por célula.

Al crear exactamente el número de células que se indiquen como dato de partida, no son admisibles niveles inferiores del dendograma con menos familias que dicho número, porque daría lugar a células vacías. El problema que se plantea en cada nivel del dendograma es equivalente al m-TSP, que busca determinar el conjunto de rutas que deben recorrer *m* comerciantes, todos ellos partiendo y volviendo a un mismo lugar. Si *m*=1, entonces se reduce al TSP si consideramos el punto de partida cualquiera de las ciudades. En tal caso, se pueden comparar las soluciones de este nuevo modelo con las investigaciones anteriores cuando el número de células es uno. La equivalencia entre los SFR y el TSP ya se comentó en las investigaciones previas (Galán, 2006), con la novedad de que existen ahora *m* células que se corresponden con los *m* comerciantes del problema m-TSP.

4.1. Métodos exactos para la resolución de los SFRC.

A continuación, mostraremos la aplicación de los métodos exactos, mediante modelos de programación lineal, de resolución del m-TSP al problema de los SFRC, de forma que podamos usar sus resultados para validar los resultados de la técnica heurística en la que nos centraremos más adelante.

4.1.1. Formulación basada en modelos de asignación.

La primera de las formulaciones basadas en los modelos de asignación, se ha utilizado en el modelo básico de SFR, desarrollado por (Galán, 2006). Este modelo equivaldría a particularizar el problema del SFRC cuando *m*=1.

Aunque no sea motivo de nuestro estudio, puede demostrarse que el número de variables y de restricciones implicadas es mayor que en la formulación basada en modelos de flujo en redes, con lo que empleará más tiempo y recursos en resolver el problema. Por este motivo, será en esta última en la que nos centremos.

4.1.2. Formulación basada en modelos de flujo en redes.

Se va a plantear como nueva investigación la utilización del modelo basado en flujo en redes para resolver los problemas de asignación de familias de productos a células de fabricación reconfigurable y secuenciación de las familias dentro de cada célula.

Modelo

$$Min \quad \sum_{l=1}^{L-C+1} \sum_{i \in F_l+\{0\}} \sum_{\substack{j \in F_l+\{0\} \\ j \neq i}} R_{ijl} \sum_{k=1}^{C} T_{ijl}^{\ k} + \sum_{l=1}^{L-C+1} K_l \sum_{i \in F_l} H_{il}$$

s.a

$$(0): \quad \sum_{l=1}^{L-C+1} K_l = 1$$

$$(1): \quad \sum_{i \in F_l+\{0\}} \sum_{k=1}^{C} T_{ijl}^{\ k} = K_l \quad : l = 1,..,L-C+1; \quad j \in F_l$$

$$(2): \quad \sum_{j \in F_l+\{0\}} T_{ijl}^{\ k} - \sum_{j \in F_l+\{0]} T_{jil}^{\ k} = 0 \quad : i \in F_l+\{0\}; \quad l = 1,..,L-C+1; \quad k = 1,..,C$$

$$(3): \quad \sum_{j \in F_l} T_{0jl}^{\ k} = K_l \quad : k = 1,..,C; \quad l = 1,..,L-C+1$$

$$(4): \quad (N_l+1) \cdot \sum_{k=1}^{C} T_{ijl}^{\ k} + U_{il} - U_{jl} \leq N_l \quad : i \in F_l; \quad j \in F_l; \quad j \neq i; \quad l = 1,..,\min(L-2; L-C+1)$$

$$(5.1): \quad T_{ijl}^{\ k} = [0,1] \quad : i \in F_l+\{0\}; \quad j \in F_l+\{0\}; \quad j \neq i; \quad l = 1,..,L-C+1; \quad k = 1,..,C$$

$$(5.2): \quad K_l = [0,1] \quad : l = 1,..,L-C+1$$

$$(5.3): \quad U_{il} \geq 0 \quad : i \in F_l; \quad l = 1,..,\min(L-2, L-C+1)$$

Datos

L: Número de niveles
(se obtiene del dendograma, donde en el nivel 1 cada familia está compuesta por 1 producto, mientras el nivel L sólo dispone de una familia que agrupa a todos los productos)
F_l: Conjunto de familias a fabricar en el nivel l, con l=1,…,L (del dendograma)
N_l: Número de familias a fabricar en el nivel l, es decir, $N_l=|F_l|$ (del dendograma)
R_{ijl}: Coste de reconfiguración al pasar de fabricar la familia i a la familia j del nivel l

$i \in F_l+\{0\} \quad j \in F_l+\{0\} \quad j \neq i \quad l = 1,..,L-1$ ()

H_{il}: Coste de no uso de recursos al fabricar la familia i del nivel l
($i \in F_l \quad l = 1,..,L$)
C: Número de células a formar

Variables

T_{ijl}^{k} = 1, si familia *i* del nivel *l* se fabrica justo antes que familia *j* del nivel *l* en célula *k*

($i \in F_l + \{0\} \quad j \in F_l + \{0\} \quad j \neq i; \quad l = 1,..,L-C+1; \quad k = 1,..,C$)

K_l = 1, si se fabrican las familias del nivel *l* (todas)

($l = 1,..,L-C+1$)

$U_{il} \geq 0$, variables auxiliares para evitar bucles de asignación de familias en cada nivel

($i \in F_l \quad l = 1,..,\min(L-2; L-C+1)$)

Objetivo

Se va a seleccionar la configuración de uno de los niveles del dendograma, asignando todas las familias de ese nivel a las células y ordenando las familias de cada célula, de forma que se minimicen los costes asociados a los cambios de familia (costes de reconfiguración) y los costes asociados a no usar los recursos o funciones dentro de cada familia (costes de no-uso).

Restricciones

(0): Hay que seleccionar un único nivel del dendograma.

(1): Paso de un viajante por cada nodo: del nivel que se seleccione, se fabricarán todas sus familias una única vez; y del resto de niveles no seleccionados, no se fabricará ninguna familia. Desde un punto de vista de un grafo, a cada familia para todas las células (excepto el nodo {0}) llegará un arco activo para todas las familias del nivel seleccionado. Al nodo {0} deberán llegar tantos arcos activos como células haya, pero eso se garantiza con las restricciones (2) y (3).

(2): Balance de flujo por nodo: si se fabrica una familia de un cierto nivel *l* en una célula *k*, entonces tendrá una familia que le preceda y otra que le siga en la secuencia; y si no se fabrica la familia, entonces no las tiene. Desde un punto de vista de un grafo, para todos los nodos incluidos el {0}, lo que entra es igual a lo que sale.

(3): Salida de viajantes del nodo inicial: en cada célula k se inicia la secuencia desde el nodo {0}. Para cada célula, saldrá un arco del nodo {0} en el nivel seleccionado.

(4): Restricciones de eliminación de subrutas: no se pueden formar subrutas dentro de cada célula. Esto no se aplica al nodo {0} sino al resto de nodos, uno por cada par de familias en todas las células.

(5): Restricciones de signo de las variables binarias (*T*, *K*) y auxiliares (*U*).

Una diferencia importante que tiene este modelo de SFRC con el de SFR es la inclusión de la familia {0}, considerada como un nodo o familia virtual, que representa la configuración inicial de partida de cada célula antes de iniciar la fabricación, y la configuración final a la que debe volver el sistema tras fabricar todas las familias asociadas a la célula. La inclusión de esta familia impondrá la necesidad de añadir unos costes de reconfiguración R_{0jl} (costes de reconfiguración desde dicha familia {0} a cada familia del dendograma) y R_{i0l} (coste de reconfiguración desde el resto de familias a la familia {0}).

Esta familia virtual es añadida para asegurar que el modelo sea válido en el nivel del dendograma superior, en el que hay tantas familias como células. En ese caso, no habría costes de no uso ($H_{il}=0$), y la solución a la que tendería el modelo es a que $T_{ijl}=0$. Como impusimos en el modelo que $j \neq i$, para dar la opción que se active, crearemos esa familia virtual (ver Figura 34).

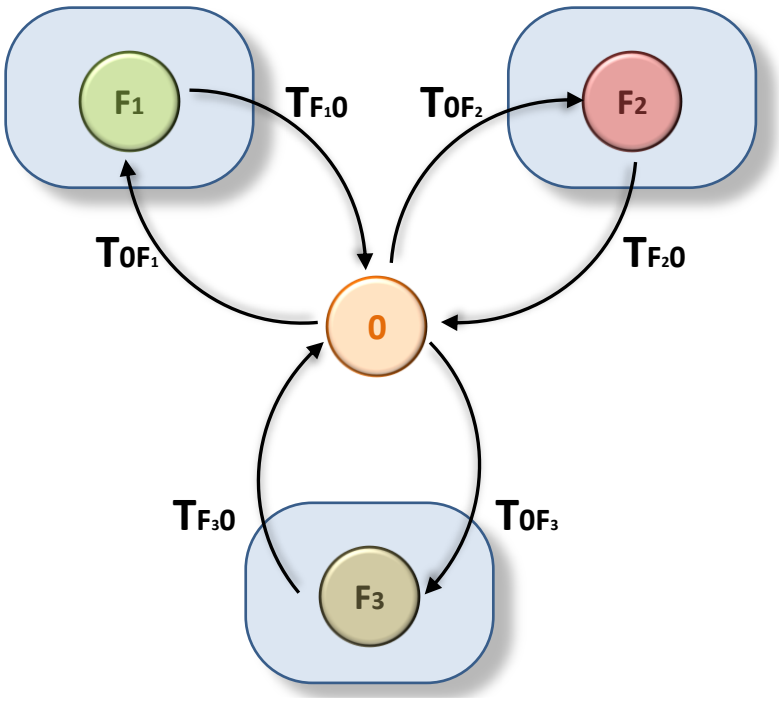

Figura 34: SFRC con F=C=3.

Se va a considerar que dichos costes R_{0jl} y R_{i0l} valgan cero. Ello conllevará que desaparezcan los costes de reconfiguración de pasar de la última a la primera familia del proceso de fabricación, con lo que los costes totales de un SFRC para el caso de que C=1 serán menores que los de un SFR planteado en las investigaciones anteriores (Galán, 2006). Esto queda ilustrado en la Figura 35.

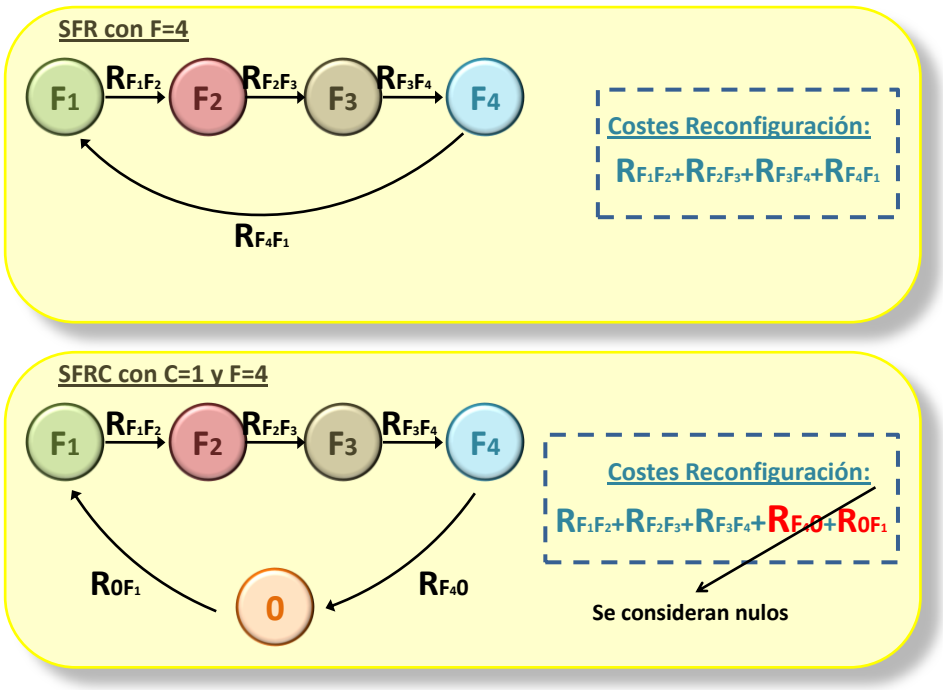

Figura 35: Comparación entre costes de reconfiguración para un SFR y un SFRC con C=1.

Debido a que el problema de los SFRC es bastante más complejo (involucra un mayor número de variables y restricciones que considerar) que el de los SFR, se demuestra que no merece la pena utilizar técnicas exactas para la resolución de problemas medianos y grandes. Esto será justificado en el siguiente capítulo, en el que veremos los resultados experimentales de las simulaciones realizadas. Por ello, se ha optado por un método aproximado, el algoritmo de la Búsqueda Tabú. Aun así, este modelo exacto basado en flujo en redes será usado para validar los resultados obtenidos.

4.2. Algoritmo de Búsqueda Tabú.

La Búsqueda Tabú es una metaheurística de mejora, una heurística genérica que parte de una solución inicial y explora soluciones vecinas en busca de una

solución mejor. Para la obtención de la solución inicial requerirá de la ayuda de otro algoritmo.

En la búsqueda de una solución óptima, permite que no nos quedemos atrapados en mínimos locales (ver Figura 36) mediante el uso de estructuras de memoria, que previenen volver a soluciones ya examinadas, o de una forma más genérica, volver hacia soluciones que compartan unos ciertos atributos con éstas (Carlton y Barnes, 1996). El uso de estas estructuras de memoria flexibles permite una realimentación, que la hace robusta frente a cambios. Esta robustez hace que esta técnica heurística resulte una opción muy interesante a la hora de resolver nuestro problema.

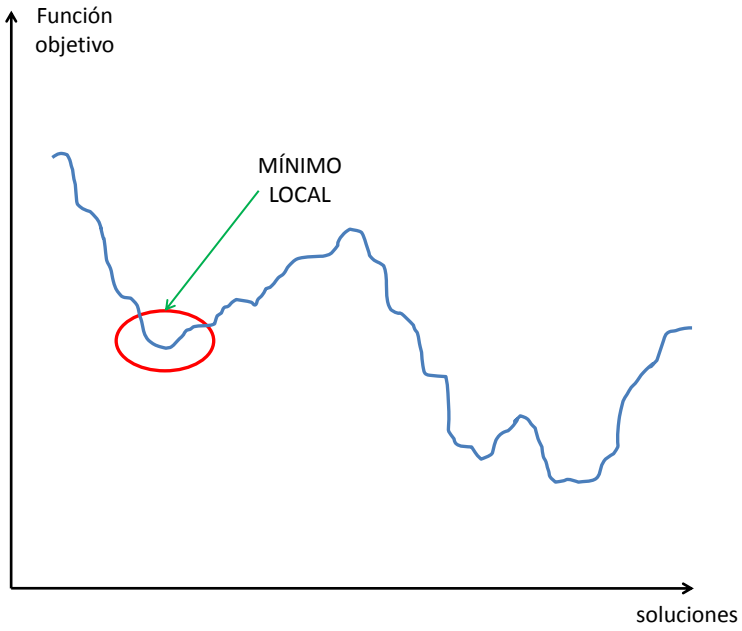

Figura 36: Mínimos locales de una Función Objetivo.

En este algoritmo, es fundamental el concepto de vecindad, o entorno, de una solución. Dada una solución, su vecindad la constituyen una serie de soluciones parecidas a ella, candidatas a ser la siguiente solución a examinar. Así, para desplazarse de una solución a otra, el algoritmo evalúa las soluciones de su vecindad y elige la mejor de todas, aunque ésta sea peor que la anterior. Este proceso volverá a repetirse un cierto número de iteraciones, hasta que se cumpla una determinada condición de parada. Para evitar que se analicen de forma cíclica las mismas soluciones, se prohíben durante un número de iteraciones la realización de movimientos que hagan volver a soluciones ya exploradas. Estos movimientos son denominados como movimientos tabú, y son los que constituyen las estructuras de memoria que requiere el algoritmo, la Lista Tabú.

Puede optarse por almacenar en la Lista Tabú las soluciones completas, en cuyo caso se denominan Listas Tabú explícitas, o bien, atributos de las soluciones, en cuyo caso se denominan Listas Tabú basadas en atributos. Además, puede darse el caso de que el algoritmo dé con una solución muy buena mediante un movimiento incluido en la Lista Tabú, por lo que se muestra necesario definir un criterio de aspiración, que permita aceptar movimientos catalogados como tabú si esto puede resultar ventajoso para la búsqueda.

Otro aspecto importante será el modo que elijamos para representar las soluciones, que afectará de manera crítica a la vecindad de una solución, ya que determinará los movimientos que pueden realizarse para pasar de una solución a su vecina, y el tamaño de la vecindad, es decir, el número de soluciones vecinas posibles.

La memoria en la que se basa la búsqueda tabú puede ser de dos tipos:

- La memoria a corto plazo es la que se encarga de crear la lista tabú, almacenando las soluciones encontradas, o sus atributos, y de evitar que estos vuelvan a producirse. Esta memoria estará determinada por la longitud de la lista tabú, que no es más que el número de iteraciones durante el que ese movimiento es considerado tabú. Además, será la encargada de controlar el criterio de aspiración y de gestionar las listas de candidatos a solución elegida, usadas para restringir el número de soluciones examinadas en una iteración determinada, bien porque el entorno es muy grande o porque la evaluación de sus elementos es muy costosa.
- La memoria a largo plazo sólo es empleada cuando la memoria a corto plazo no ha proporcionado buenas soluciones. Este tipo de memoria también se denomina memoria basada en frecuencia, ya que dependerá de la frecuencia con la que se hayan presentado ciertos atributos de las soluciones encontradas. Podemos distinguir dos tipos de memoria a largo plazo:

    - La estrategia de intensificación favorece los movimientos que hayan sido buenos en el pasado, es decir, soluciones con atributos que hayan aparecido múltiples veces. Lo más común, es reiniciar la búsqueda a partir de la mejor solución actual, y mantener fijos los atributos que parecen mejores.

- La estrategia de diversificación trata de buscar mejores soluciones en regiones poco exploradas, es decir, con atributos que han aparecido poco en las mejores soluciones encontradas.

Podemos resumir el funcionamiento de este algoritmo mediante el diagrama de flujo de la Figura 37:

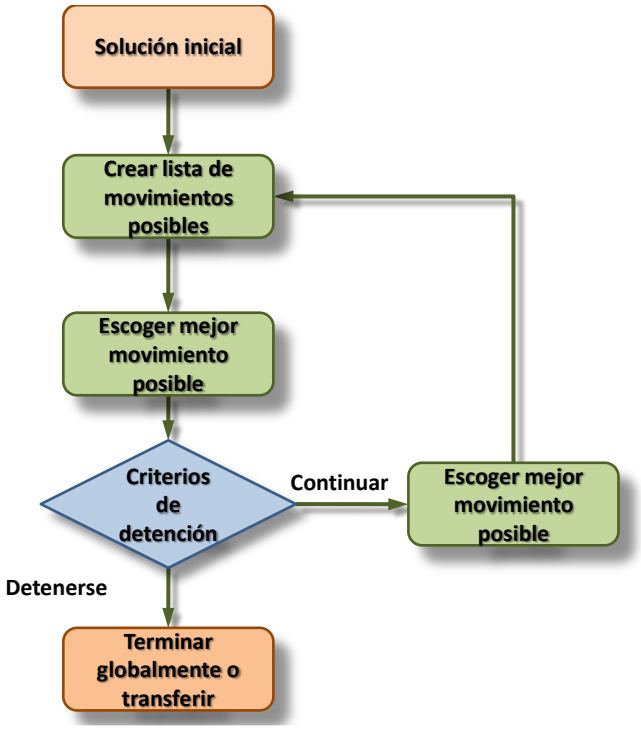

Figura 37: Diagrama de flujo de la Búsqueda Tabú.

Al ser la elección del mejor movimiento posible un paso crítico en el algoritmo, profundizaremos en él mediante la Figura 38:

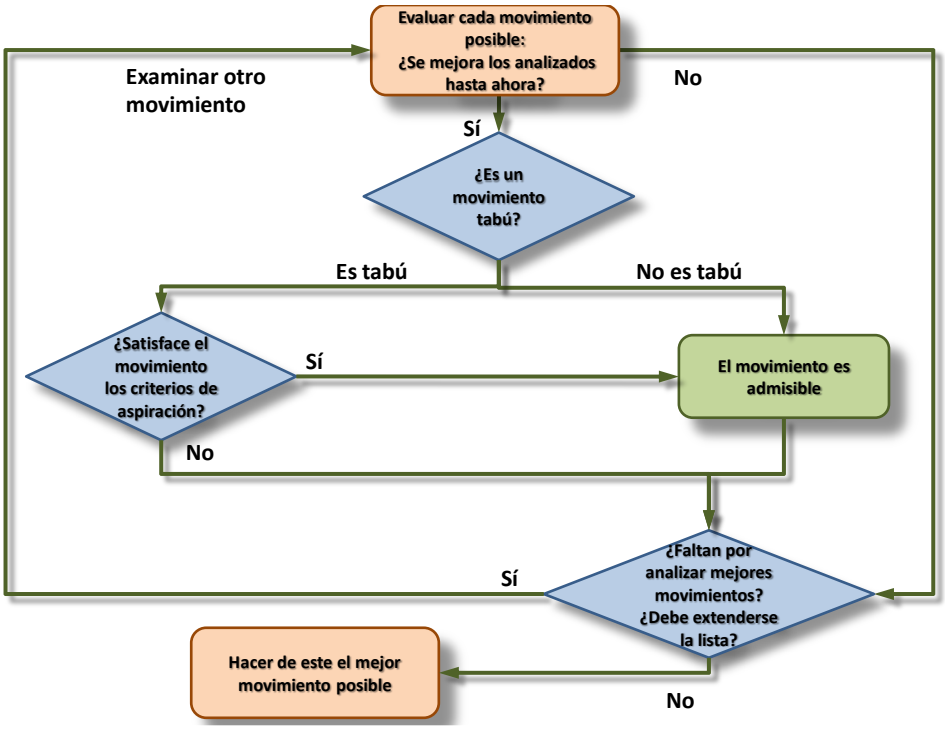

Figura 38: Elección del mejor movimiento posible.

4.3. Adaptación de la Búsqueda Tabú a nuestro problema de SFRC.

A continuación, iremos detallando los elementos de la búsqueda tabú utilizados en la resolución de nuestro problema, el de selección y secuenciación de familias de productos en un SFRC.

4.3.1. Representación de las soluciones.

El primer paso del algoritmo de búsqueda tabú consiste en representar cada solución del problema. En dicha representación deben aparecer qué familias de productos se fabrican en cada célula, y en qué orden se producen.

Hay múltiples formas de representar estas soluciones mediante el uso de estructuras de memoria, como matrices o vectores. Las matrices emplean mayor cantidad de memoria, por lo que nos centraremos en el estudio de cómo representar nuestra solución mediante vectores.

En (Carter y Ragsdale, 2006) se proponen tres posibles representaciones para resolver el m-TSP, si bien ellos lo hacen con la finalidad de aplicarlos a otra metaheurística, los algoritmos genéticos.

Para ilustrar estas representaciones, partiremos de un sistema en el que se fabrican un número de familias de productos *F*=10, en *C*=3 células. Cada familia será identificada por un número secuencial. La solución que representaremos mediante las distintas soluciones será la del caso en el que en la primera célula fabricásemos la secuencia de familias {5-8-4}, en la segunda la {2-1-0} y en la tercera las familias {9- 3-7-6}.

- En la primera representación, empleamos un vector de tamaño *F+C-1*, en el que tenemos *F* familias dispuestas en *C* células (ver Figura 39). Para separar la secuencia de productos que se fabrican en cada célula emplearemos números enteros negativos.

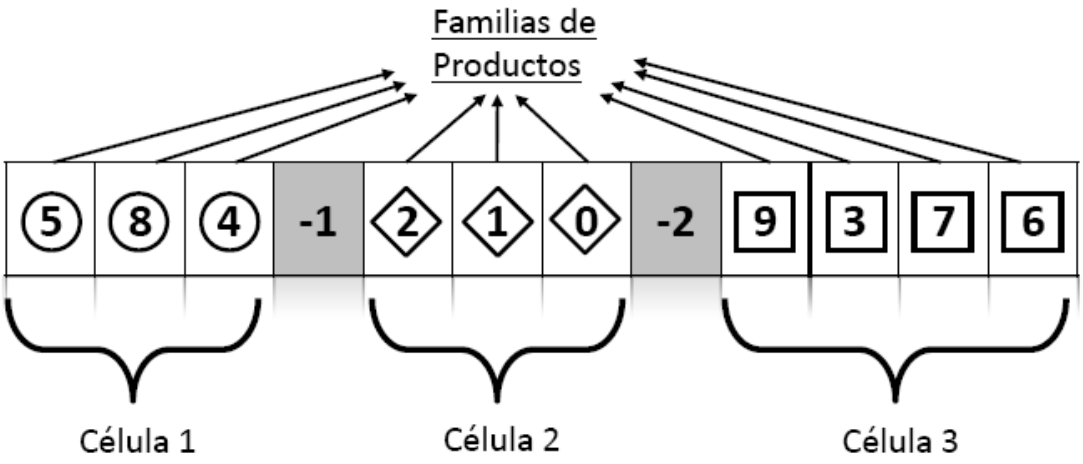

Figura 39: Primera representación de una solución del m-TSP.

- En la segunda representación, empleamos dos vectores distintos, ambos de tamaño *F*. Como podemos ver en la Figura 40, en el primer vector representamos las familias, mientras que el segundo representamos la célula en la que se fabrica la familia que está en su misma posición en el primer vector (Malmborg, 1996 y Park, 2001).

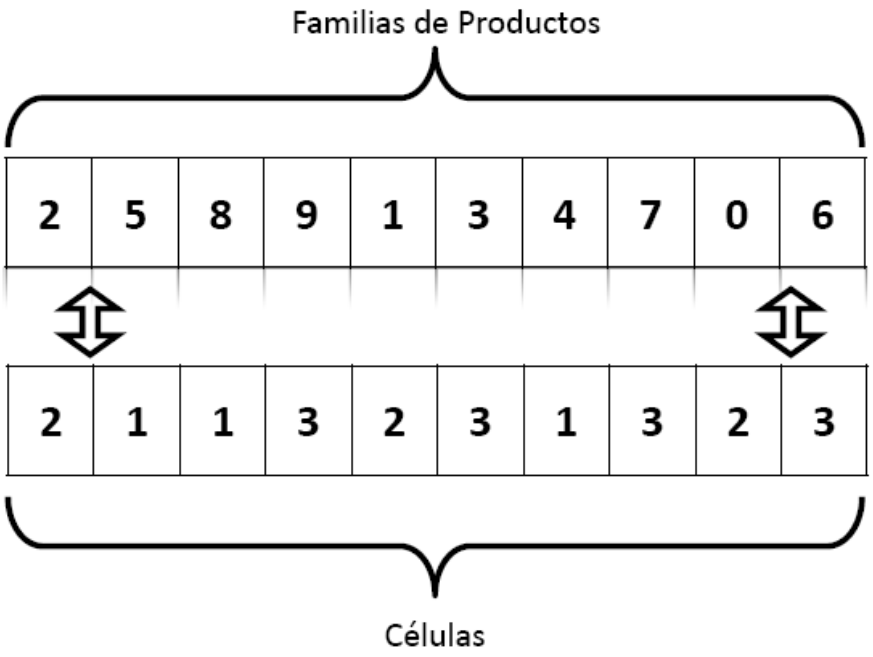

Figura 40: Segunda representación de una solución del m-TSP.

- En la tercera representación, se pretende compactar la información que proporciona la segunda representación en un único vector, de tamaño *F+C*, en el que aparecen dos partes diferenciadas. La primera parte, de tamaño *F*, representa las secuencias de familias, mientras que la segunda, de tamaño *C* representa el número de familias asignada a cada una de las células (ver Figura 41).

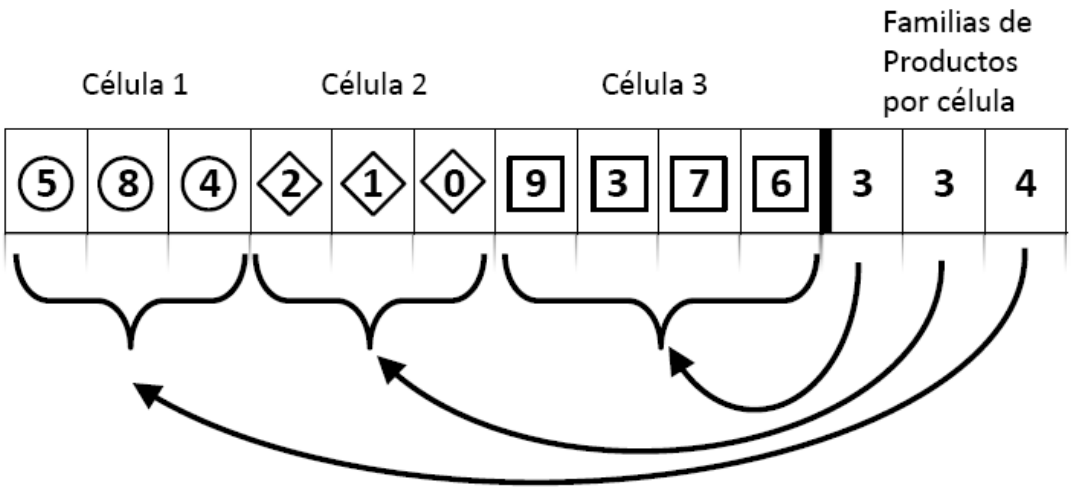

Figura 41: Tercera representación de una solución del m-TSP.

Como también es razonado en (Carter y Ragsdale, 2006), esta representación de las soluciones reduce el espacio posible de soluciones, dado que elimina muchas soluciones redundantes (aunque, no todas). Esto

hace que la solución sea más compacta, por lo que será la utilicemos en nuestra implementación del algoritmo.

4.3.2. Generación de la solución inicial admisible.

Para obtener una solución inicial admisible, se aplica un método en dos fases. Primero, emplearemos una heurística específica para determinar la secuencia de fabricación de las familias de productos. En segundo lugar, dispondremos estas familias de productos en las distintas células.

Para determinar el orden en el que deben fabricarse las familias, emplearemos una heurística muy simple de implementar, variante de la heurística del camino mínimo (Rosenkrantz *et al.*, 1977). En ésta, en vez de seleccionar la primera familia de forma aleatoria, se analizarán los costes de reconfiguración de pasar de una familia a otra, seleccionando el par que provea un menor coste. Una vez seleccionado el primer par de familias a fabricar, se va construyendo la secuencia añadiendo la familia con el coste de reconfiguración mínimo. El procedimiento finaliza cuando todas las familias están en la secuencia de producción (Racero *et al.*, 2007). Para ello no es necesario utilizar los costes de no-uso, ya que permanecen constantes independientemente de la secuencia de fabricación.

Una vez determinado el orden de producción de las familias, se reparten entre las células dividiendo el número de familias a fabricar entre el número de células, y asignando secuencialmente las familias a cada célula en el mismo orden anterior. De esta manera ninguna célula se quedará vacía. Es decir, se genera una secuencia con todas las familias, formando la primera parte del vector que representa una solución, y se reparten las F familias entre las C células, obteniendo los valores de la segunda parte del vector.

4.3.3. Movimientos y soluciones vecinas.

Dos soluciones son vecinas entre sí si son muy similares, de tal forma que partiendo de una de ellas podemos llegar a la otra mediante un único movimiento. Dependiendo de los movimientos que definamos para movernos de una solución a otra, la vecindad de una solución será de un tamaño u otro.

Se van a distinguir dos grupos de movimientos para generar las soluciones vecinas de una solución, los movimientos intracelulares y los movimientos intercelulares:

- Los movimientos intracelulares son movimientos que no alteran las familias que componen cada célula, sólo alteran la secuencia en la que se fabrican dentro de cada célula. Se distinguen dos tipos de movimientos intracelulares, los movimientos "SWAP" e "INSERT":

    - El movimiento "SWAP" consiste en el intercambio de las posiciones de dos familias de una misma célula (ver Figura 42).

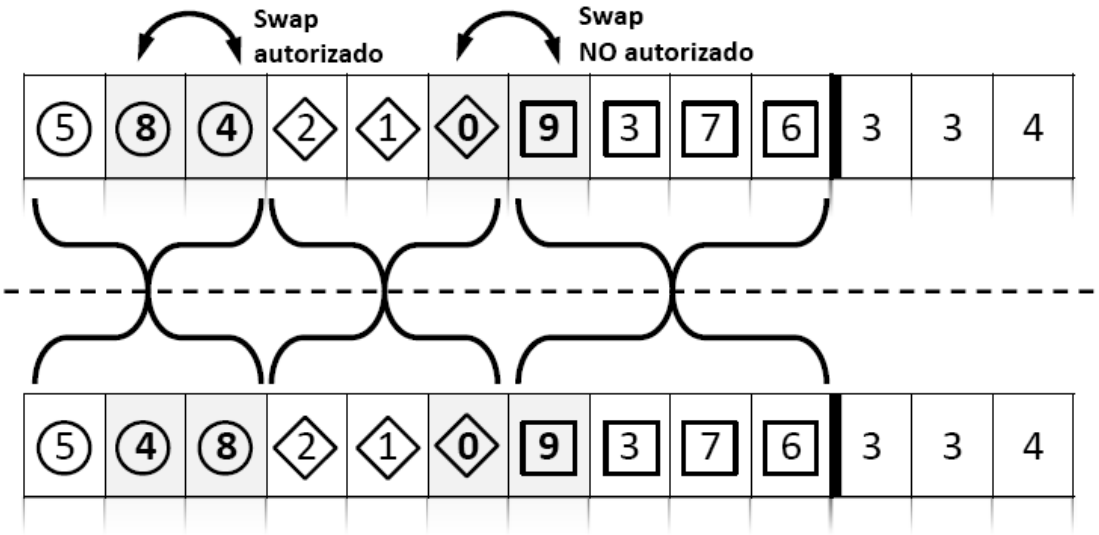

Figura 42: Movimiento "SWAP" intracelular.

    - El movimiento "INSERT" consiste en la inserción de una familia en la posición que ocupa otra dentro de la misma célula (ver Figura 43).

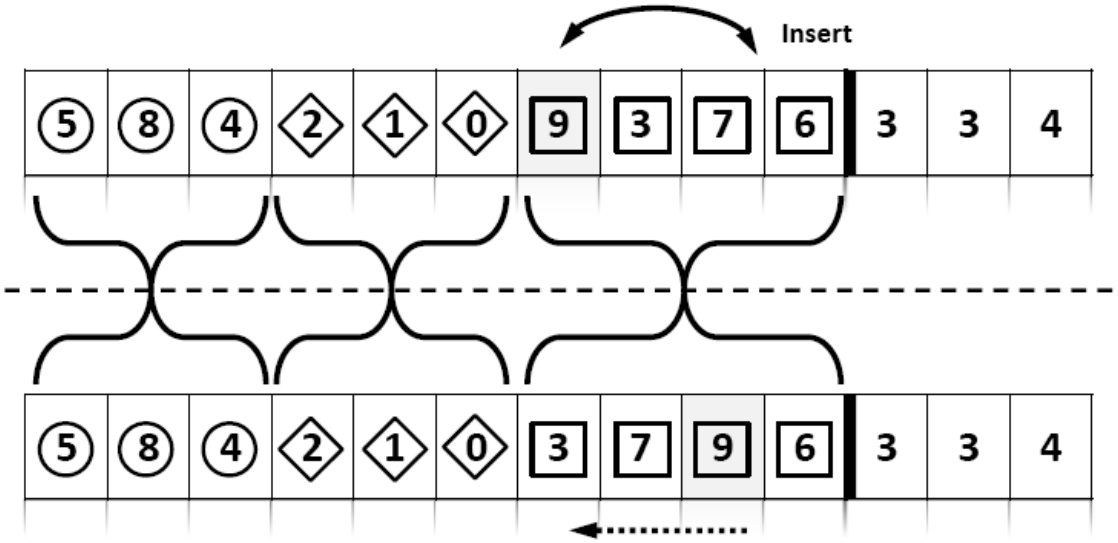

Figura 43: Movimiento "INSERT" intracelular.

- Los movimientos intercelulares son movimientos en los que cambian las familias que componen las células, pudiendo modificar el tamaño de las mismas. Al igual que en los movimientos anteriores, podemos definir los movimientos intercelulares "SWAP" e "INSERT", con la única diferencia de que involucran a familias de células distintas.

    - El movimiento "INTER-SWAP" consiste en el intercambio de las posiciones de dos familias pertenecientes a células distintas, sin que se modifique el tamaño de las mismas (ver Figura 44).

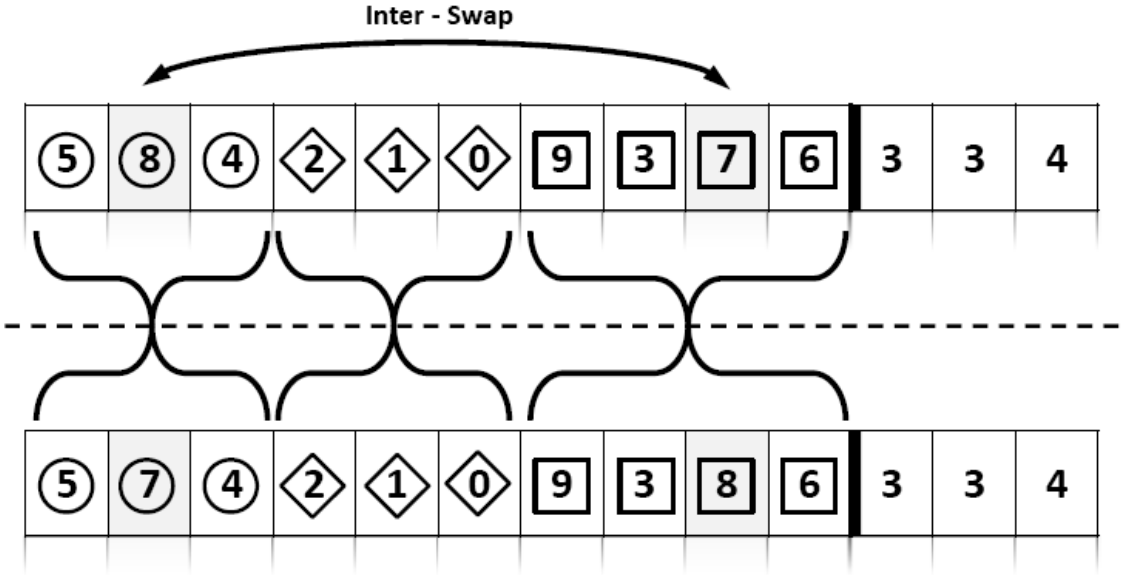

Figura 44: Movimiento "SWAP" intercelular (INTER-SWAP).

- El movimiento “INTER-INSERT” consiste en la inserción de una familia en la posición que ocupa otra familia de otra célula, modificando el tamaño de ambas células (ver Figura 45). Este movimiento no se permite a células de una sola familia, para evitar que ésta se vacíe.

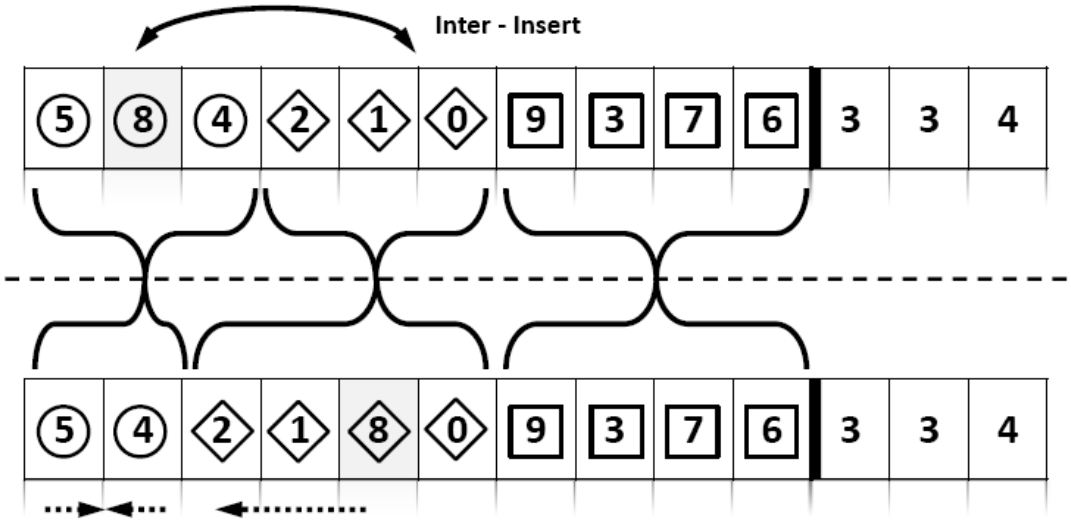

Figura 45: Movimiento “INSERT” intercelular (“INTER-INSERT”).

Si se exploran todas las soluciones vecinas a partir de los movimientos anteriores, tendremos unas vecindades con el tamaño mostrado en la Tabla 3. Consideramos que C_i es el tamaño de la célula *i*-ésima, C el número de células y F el número de familias, que equivale a la suma de los C_i.

SWAP	$\frac{C_1 \cdot (C_1 - 1)}{2} + \ldots + \frac{C_C \cdot (C_C - 1)}{2} = \frac{\sum_{i=1}^{C} C_i^2 - \sum_{i=1}^{C} C_i}{2} = \frac{\sum_{i=1}^{C} C_i^2 - F}{2}$
INSERT	$C_1 \cdot (C_1 - 1) + \ldots + C_C \cdot (C_C - 1) = \sum_{i=1}^{C} C_i^2 - \sum_{i=1}^{C} C_i = \sum_{i=1}^{C} C_i^2 - F$
INTER-SWAP	$\frac{C_1 \cdot (F - C_1)}{2} + \ldots + \frac{C_C \cdot (F - C_C)}{2} = \frac{F \cdot (\sum_{i=1}^{C} C_i) - \sum_{i=1}^{C} C_i^2}{2} = \frac{F^2 - \sum_{i=1}^{C} C_i^2}{2}$
INTER-INSERT	$\leq C_1 \cdot (F - C_1 + C - 1) + \ldots =$ $(\sum_{i=1}^{C} C_i)^2 + C \cdot \sum_{i=1}^{C} C_i - \sum_{i=1}^{C} C_i - \sum_{i=1}^{C} C_i^2 = F^2 + F \cdot (C - 1) - \sum_{i=1}^{C} C_i^2$

Tabla 3: Tamaños de las vecindades.

Como puede observarse, los tamaños de las vecindades, especialmente en el caso de los movimientos intercelulares, son elevados cuando son grandes los valores de *C* y *F*, y puede ser interesante reducir dicho tamaño a través de un subconjunto de los movimientos posibles, para así disminuir el tiempo de generación de una nueva solución.

- Una variante del movimiento "INTER-SWAP", denominado "INTER-SWAP-MISMA", consistiría en intercambiar las posiciones sólo entre familias de células distintas que ocupen la misma posición en la secuencia de fabricación en su célula correspondiente.
- Una variante del movimiento "INTER-INSERT", denominado "INTER-INSERT-MISMA", consistiría en sólo insertar las familias de las células en la posición final del resto de las células.

Los movimientos intracelulares ("SWAP" e "INSERT") y el movimiento "INTER-SWAP" (o su variante) no modifican los tamaños de las células, por lo que no pueden abarcar todo el espacio de soluciones. Por este motivo se hace necesario combinar estos movimientos con el movimiento "INTER-INSERT" (o su variante). Se han realizado experimentos combinando varios de los movimientos, cuyos resultados mostraremos en el siguiente capítulo.

4.3.4. Lista tabú: atributos y tamaño.

Como ya hemos mencionado con anterioridad, en la lista tabú podemos almacenar las últimas soluciones exploradas, o una serie de atributos que las describan. Este último caso es más óptimo, ya que evita un uso excesivo de memoria, y acelera el funcionamiento del algoritmo.

En la Figura 46 y en la Figura 47 podemos ver los atributos que almacenaremos:

- Un atributo común para todos los movimientos analizados será un entero que identifique de qué tipo de movimiento se trata.
- Además, en el caso de los movimientos intracelulares hay que codificar la célula involucrada, mediante un número entero, mientras que en el caso de los movimientos intercelulares necesitaremos almacenar las dos células involucradas en el movimiento.
- Los atributos restantes dependerán del tipo de movimiento:

- En el “SWAP”, se almacenan las dos familias involucradas en el intercambio.
- En el “INSERT”, se almacenan las posiciones en las que se encuentran las familias involucradas en el movimiento, la posición donde se encuentra la familia origen y la posición destino de ésta. Es importante destacar que no es válido almacenar, tal y como se hace en el “SWAP”, las familias involucradas, ya que tras el movimiento de “INSERT” cambia la topología de la solución.
- En el “INTER-SWAP”, se almacenan las familias intercambiadas. En la variante mencionada anteriormente, bastaría con almacenar la posición común de intercambio.
- En el caso del “INTER-INSERT”, se almacenan los mismos atributos que en el “INSERT” intracelular. En la variante mencionada anteriormente, sólo necesitaríamos almacenar la familia origen o la posición en la que se encuentra.

El tamaño de la Lista Tabú está relacionado con la memoria a corto plazo que tiene el algoritmo. En la Lista Tabú se almacenan los atributos de las mejores soluciones vecinas que más recientes, prohibiendo volver a esas soluciones, lo que ocasionaría volver a analizar soluciones ya exploradas. Si la lista tiene un tamaño demasiado pequeño, la Búsqueda Tabú puede regresar al mismo óptimo local, impidiendo la exploración de un área más amplia del espacio de soluciones. Si el tamaño de la lista es muy grande, se necesitará demasiado tiempo para recorrer toda la vecindad, de forma que ralentizaría mucho el algoritmo determinar si un movimiento está ya almacenado en la Lista Tabú.

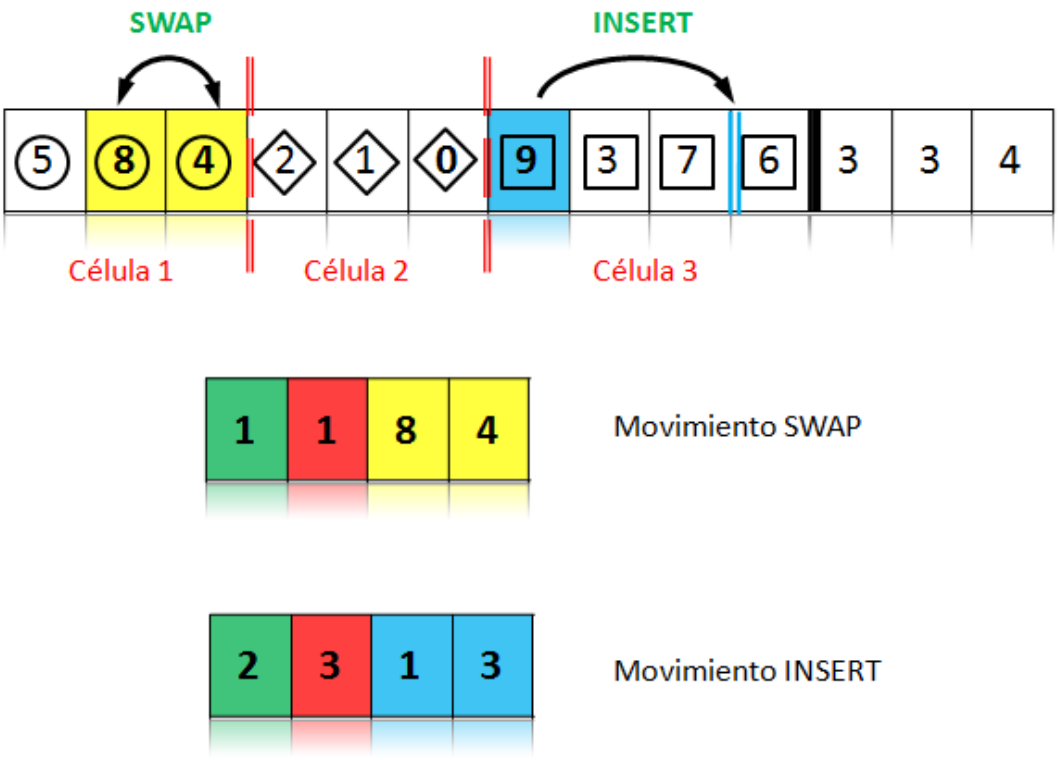

Figura 46: Representación de los movimientos intracelulares.

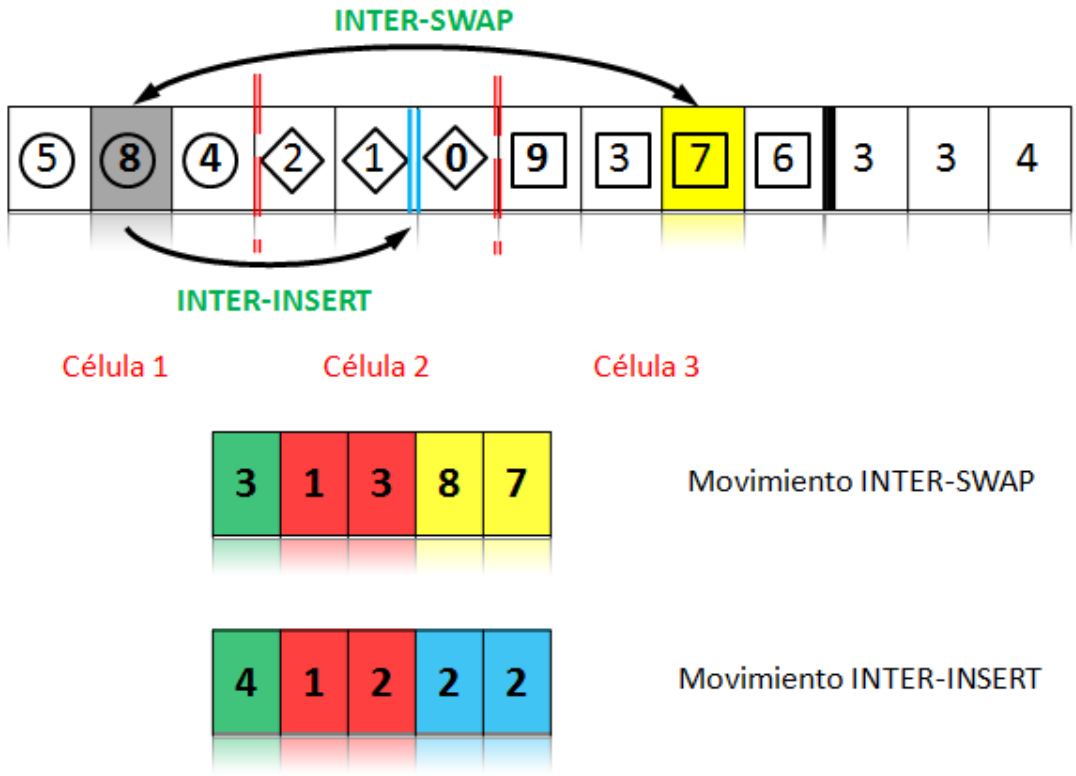

Figura 47: Representación de movimientos intercelulares.

4.3.5. Evaluación de los costes de cada solución.

Como ya hemos incidido en múltiples ocasiones, lo que determina que una solución sea menor que otra es que implique un menor coste. Sólo mencionar, que dado que validaremos los resultados de nuestro algoritmo con los del modelo de programación lineal, y éste no considera los costes de reconfiguración de la última familia a la primera del sistema productivo (debido a la inclusión de la familia virtual {0}), a la hora de calcular los costes de una solución generada por nuestro algoritmo, tampoco deberemos tener estos costes en cuenta. Esto queda ilustrado en la Figura 48, para el caso de una única célula. Obviamos los costes de no-uso asociadas a la familia de la secuencia, ya que éstos no se verán afectados por este cambio.

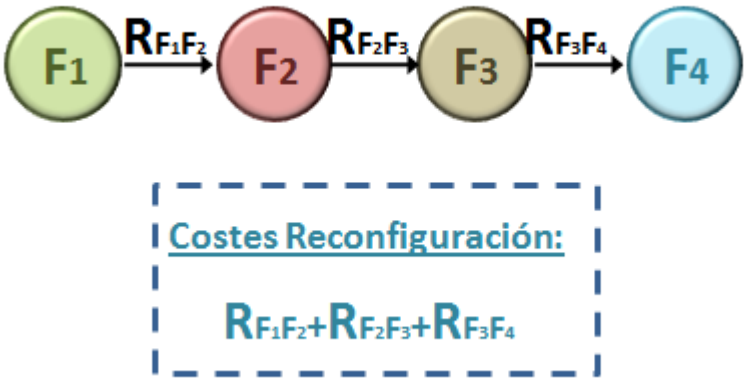

Figura 48: Cálculo de costes de reconfiguración para un SFRCcon C=1 y F=4.

4.3.6. Criterio de finalización.

Podemos considerar diversos criterios de finalización, como, la realización de un número fijo de iteraciones (n_{total}), o la realización de un determinado número de iteraciones (n_{ism}) sin mejorar la solución encontrada.

Estos criterios pueden ser usados simultáneamente, siempre que se dé que $n_{total}>n_{ism}$. El número n_{ism} se expresa como un porcentaje del n_{total}. Dichos criterios pueden ser obtenidos mediante experimentación.