

## Apéndice E. Catálogo de operadores de vecindad

### Operador 1 – Modificar niveles de producción

Este es un operador simple que busca introducir variaciones en el inventario global disponible en el sistema. Se elige un periodo  $t$  a modificar, y la producción de ese periodo se multiplica por un factor aleatorio entre 0.8 y 1.2 con el fin de explorar si ese cambio en la producción permite reducir costos de almacenamiento, siempre y cuando sea posible seguir abasteciendo la demanda de los clientes. Para evidenciar este operador y los demás de forma gráfica, se plantea el siguiente ejemplo.

#### Tabla

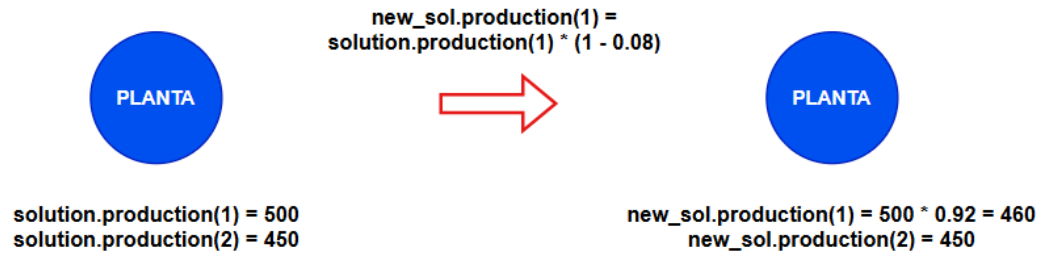
*Parámetros del ejemplo ilustrativo*

| Parámetro                 | Definición                       | Valor                       |
|---------------------------|----------------------------------|-----------------------------|
| <b>D</b>                  | Número de distribuidores         | 2                           |
| <b>C</b>                  | Número de clientes               | 2                           |
| <b>K</b>                  | Tamaño flota de vehículos $k$    | 2                           |
| <b>M</b>                  | Tamaño flota de vehículos $m$    | 4                           |
| <b><math>Q_k</math></b>   | Capacidad flota de vehículos $k$ | 500                         |
| <b><math>Q_m</math></b>   | Capacidad flota de vehículos $m$ | 150                         |
| <b>T</b>                  | Número de periodos               | 2                           |
| <b><math>dem_u</math></b> | Demanda del cliente $u \in C$    | En el intervalo<br>[70, 90] |

La siguiente figura permite observar el funcionamiento de este proceso, en donde se selecciona el periodo  $t = 1$  para modificar y se realiza un ajuste aleatorio definido por  $\pm 20\%$ .

### Figura

#### *Modificación de solución vecina – Producción en planta*

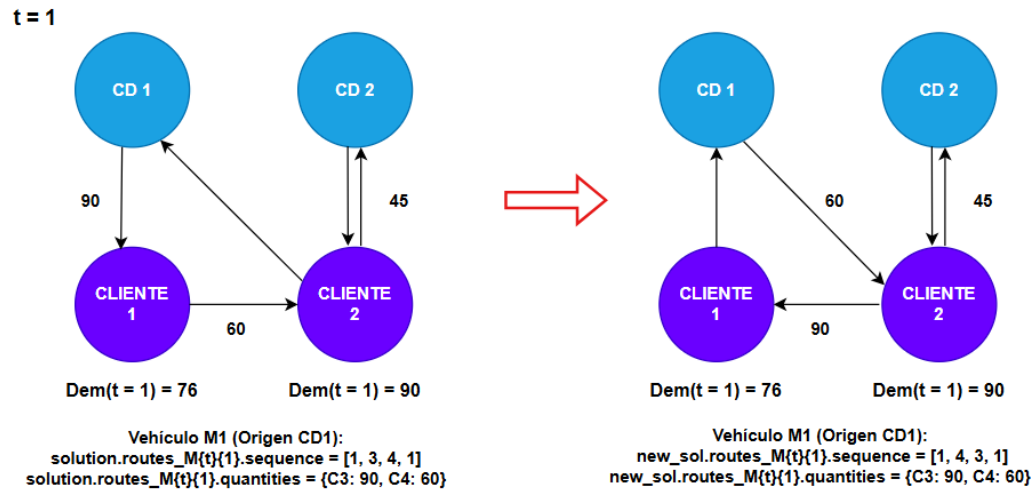


### Operador 2 – Intercambio de clientes intra-ruta (Swap)

Este operador busca optimizar el orden de visita a los clientes dentro de una ruta ya existente. Los clientes que se visitan en una ruta no cambian, pero si la secuencia en que son visitados estos nodos. Para esto se elige un periodo a modificar al azar, y se escoge una ruta que tenga al menos dos clientes, la secuencia dentro de esa ruta se extrae y se seleccionan dos posiciones distintas al azar dentro de esa secuencia, con el fin de evaluar si este cambio representa una disminución en los costos asociados al ruteo. La siguiente figura evidencia este proceso. En esta, solo se describe el ruteo del vehículo de interés ( $m = 1$  en este caso), a pesar de que se use más de un vehículo en la solución.

**Figura 1**

*Modificación de solución vecina – Intercambio de clientes Intra-Ruta*



### Operador 3 – Intercambio de clientes inter-ruta

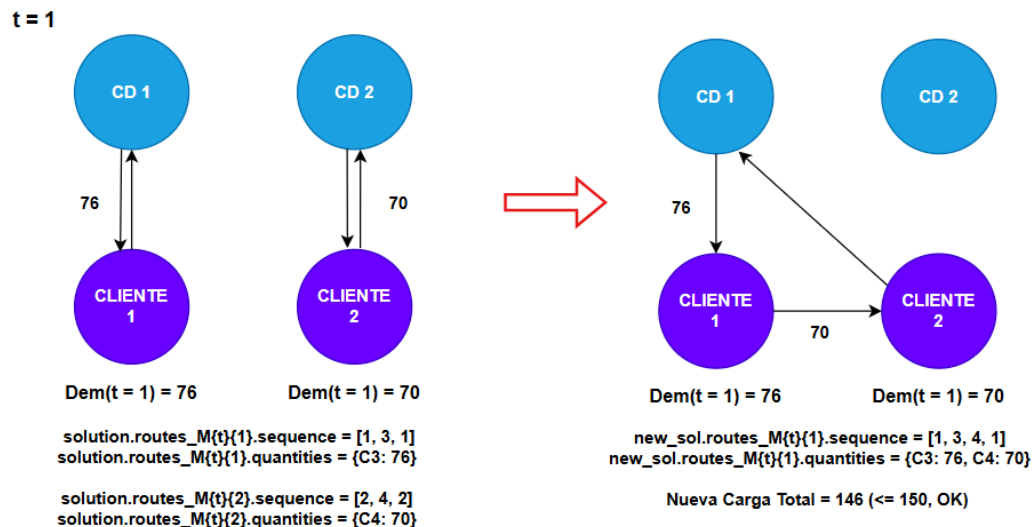
Este operador es similar al anterior pero actúa de forma más disruptiva, ya que su objetivo es explorar si los costos de ruteo pueden disminuir moviendo un cliente de una ruta a otra. Al ser un operador de mayor complejidad, algunos ajustes fueron necesarios a la hora de trabajar con instancias de mayor tamaño, principalmente porque el aumento en la cantidad de nodos representa un incremento desmesurado en el requerimiento computacional. Todos los ajustes realizados en este operador (y en los demás) quedan debidamente plasmados en este documento.

Para las instancias pequeñas, se elige un cliente al azar de la ruta “M\_A”, se elimina y se elige una nueva ruta de destino “M\_B” completamente al azar entre las rutas generadas en ese periodo. El problema al escalar a las instancias grandes reside en que muchas veces, al pasar una ruta “M\_A” que parte de un CD a otra “M\_B” que parte de otro CD, muy comúnmente no había suficiente inventario en ese nuevo CD de origen para hacer un cambio

de ese estilo, generando soluciones no factibles constantemente. Por esto, la diferencia fundamental radica en que para las instancias grandes, en lugar de buscar en todas las rutas creadas, el operador se enfoca únicamente en las que también parten del mismo CD de origen. Y sólo se realiza el movimiento si la cantidad a mover del cliente cabe en la ruta de destino sin exceder la capacidad del vehículo. En la figura que se muestra a continuación se presenta un ejemplo en donde el cliente 2 se inserta en la ruta  $m = 1$ , y la ruta  $m = 2$  queda vacía.

### Figura

*Modificación de solución vecina – Intercambio de clientes Inter-Ruta*



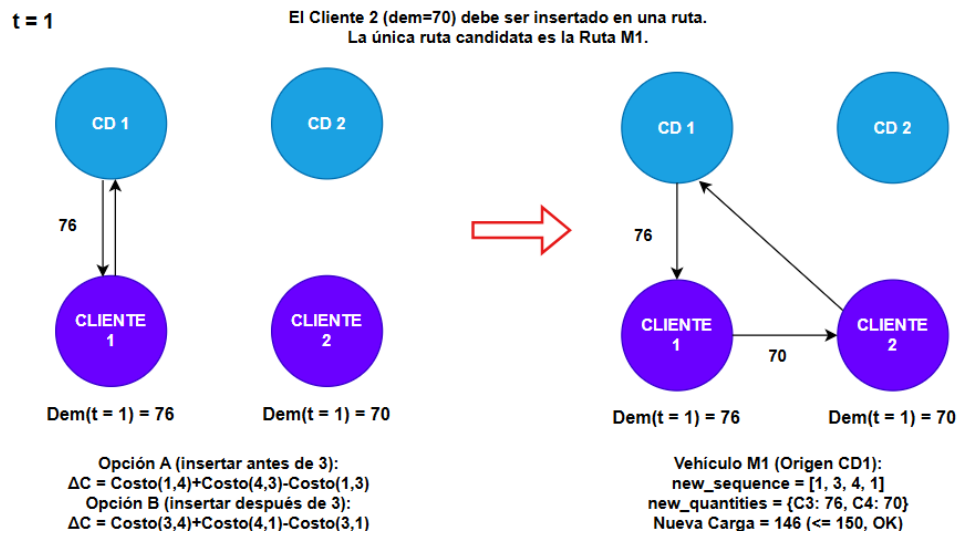
### Operador 4 – Inserción de menor costo (Best Insertion)

Este operador funciona de forma similar al anterior, pero con una estrategia más inteligente y deliberativa. Básicamente intenta encontrar el mejor lugar posible dentro de todas las opciones para hacer la inserción de un cliente de una ruta a otra. Primero, al igual que en el operador 3, se selecciona un cliente de forma aleatoria y una vez hecho esto, se inicia una búsqueda sistemática donde se itera sobre todas las rutas del sistema. Una vez confirmado que el vehículo  $m$  tiene suficiente capacidad para recibir al nuevo cliente, se itera

sobre todas las posiciones de inserción posibles dentro de esa ruta. Para cada posición, se calcula el costo marginal con la variable “delta\_costo” y se selecciona la inserción que represente una disminución en los costos de transporte. La diferencia con el operador para las instancias más grandes es que se hace el ajuste para que solo se evalúen las rutas que parten del mismo CD de origen y la verificación de capacidad del vehículo es más completa, permitiendo que solo se obtengan soluciones vecinas factibles.

## Figura

### *Modificación de solución vecina – Inserción de menor costo*



## Operador 5 – Algoritmo 2-Opt

Este operador se deriva de un algoritmo clásico y potente utilizado principalmente en otros problemas como el TSP. Su objetivo es identificar y eliminar cruces de caminos que puedan generar ineficiencias o sobrecostos. En esencia, este algoritmo busca mejorar rutas ya creadas intercambiando dos nodos que no son adyacentes, minimizando así distancias de viaje. Debido a que visualmente es muy similar al operador 2 con un sistema con solo dos clientes, su ejecución se explica a continuación, en un sistema con 3 clientes.

Inicialmente, se elige de forma aleatoria una ruta del segundo escalón en la que se visite a más de un cliente. Después, se seleccionan las aristas dirigidas no adyacentes que forman esa ruta y se realiza el intercambio. Una vez realizado, se compara el costo de la configuración antigua con la actual para determinar si el intercambio es beneficioso. Esta situación se ejemplifica seguidamente.

Por ejemplo, para una ruta  $CD_1 \rightarrow C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow CD_1$  se seleccionan el conjunto de aristas dirigidas  $(C_1, C_2)$  y  $(C_3, CD_1)$ . Después de esto, se evalúa el intercambio:

Costo actual:  $\text{Costo}(C_1, C_2) + \text{Costo}(C_3, CD_1)$

Costo 2-Opt:  $\text{Costo}(C_1, C_3) + \text{Costo}(C_2, CD_1)$

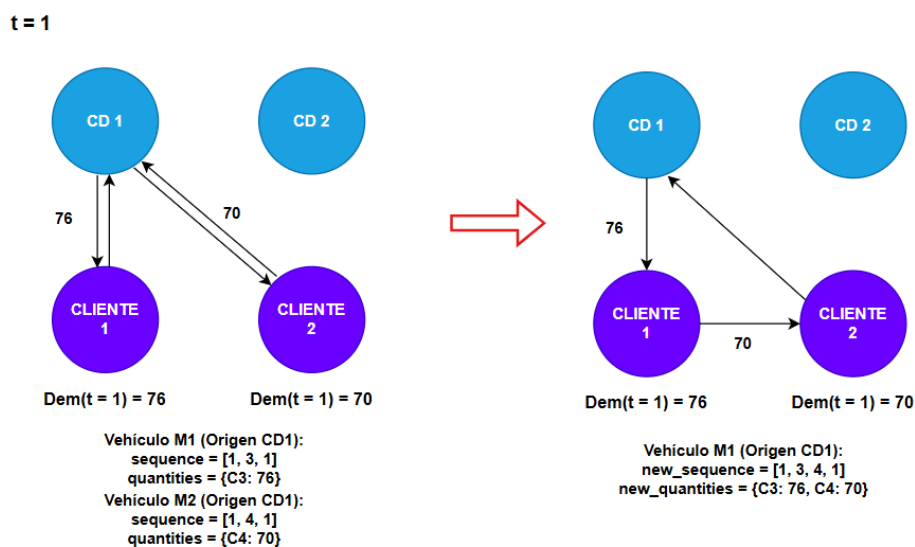
Si el costo nuevo obtenido por 2-Opt es menor al costo actual, el intercambio es beneficioso y el algoritmo procede a ejecutarlo, por lo que la ruta final sería  $CD_1 \rightarrow C_1 \rightarrow C_3 \rightarrow C_2 \rightarrow CD_1$ .

#### **Operador 6 – Fusionar rutas M del mismo origen**

Este operador se encarga de fusionar rutas cuando dos o más vehículos salen del mismo CD, esto con el objetivo de disminuir costos de envío, convirtiendo estrategias de transporte directos en rutas consolidadas en un mismo viaje. Al fusionar esas rutas, se busca reducir costos y mejorar la utilización de la capacidad de cada vehículo m. Antes de tomar la decisión de fusionar dos rutas se realiza un chequeo de capacidad, si este chequeo es exitoso se fusionan las rutas y el contenido de la ruta M\_B se anexa al de la ruta M\_A. La figura a continuación muestra este proceso de forma gráfica.

## Figura

### *Modificación de solución vecina – Fusión de rutas M*

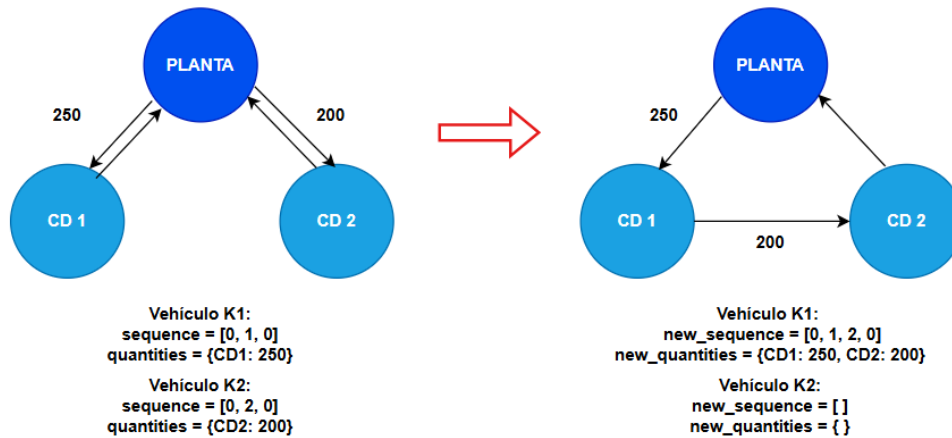


## Operador 7 – Fusionar rutas K (primer escalón)

Este es un operador análogo al operador 6, pero enfocado en mejorar la eficiencia del ruteo en el primer escalón. Este operador es particularmente útil debido a la gran capacidad de los vehículos k, lo que permite atender a varios CDs sin necesidad de tener que utilizar muchos vehículos de la flota. Nuevamente, se realiza un chequeo de capacidad antes de tomar la decisión de fusionar envíos, si la carga que se quiere combinar es menor a la capacidad disponible del vehículo, se ejecuta la fusión.

## Figura

### *Modificación de solución vecina – Fusión de rutas K*



### Operador 8 – Redirección estratégica de rutas K (primer escalón)

Este operador busca consolidar más volumen de envío hacia el CD que es más económico abastecer desde la planta. Para esto, el algoritmo analiza la matriz de costos  $c1$  e identifica el CD cuyo costo de envío es menor. Luego, el algoritmo elige alguna ruta  $k$  (preferiblemente una que no esté sirviendo al CD estratégico) y ejecuta la redirección de toda su carga hacia el nuevo CD. Para las instancias grandes es necesario relajar este operador, ya que redirigir todo el flujo a un único CD casi siempre convierte a ese vecino en una solución inviable, debido a la gran cantidad de clientes a abastecer. Con base en esto, en las instancias de mayor envergadura el operador modifica una sola ruta a la vez, aumentando la posibilidad de generar vecinos factibles. Este operador es bastante disruptivo por lo que es posible que se creen desbalances de inventario, lo que finalmente es evaluado por *evaluateSolution* para verificar su factibilidad.

### Operador 9 – Redirección estratégica de rutas M (segundo escalón)

Este operador trabaja a partir de las mismas bases del operador 8, pero con un enfoque en el segundo escalón. Para esto, se sigue tomando el CD estratégico encontrado anteriormente y se selecciona la ruta a modificar que tenga origen en un CD diferente al que



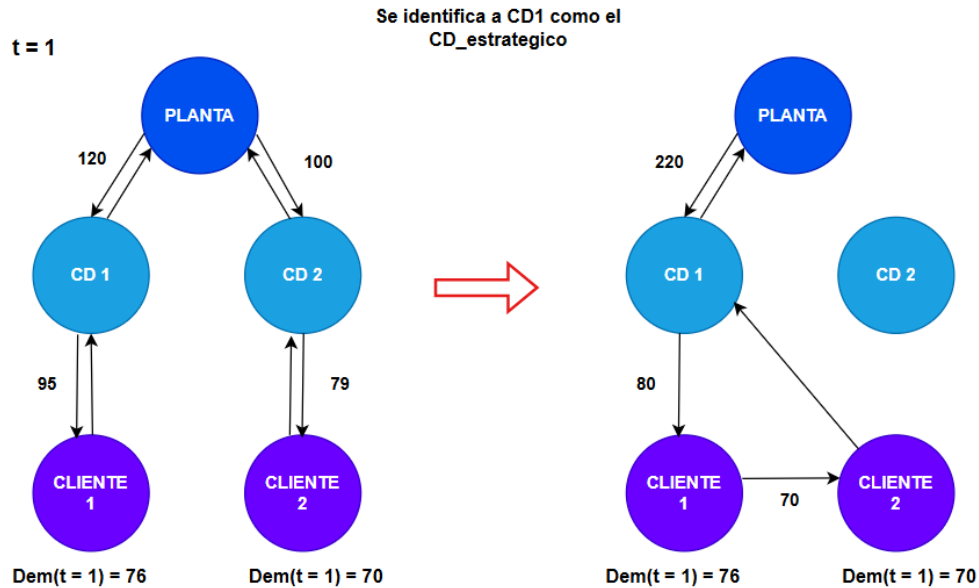
se identificó como estratégico. Este operador va de la mano con el anterior y permite que ese desbalance de inventario al que se pudo llegar tenga una solución eficiente, en donde el enfoque ahora esté en la distribución de inventario desde el CD que recibe más producto desde la planta. Para las instancias pequeñas, la idea original del operador es reconstruir todas las rutas  $M$  para la reducción de costos. Este proceso es sumamente costoso en términos computacionales en las instancias con una gran cantidad de clientes, por lo que solo se modifica una ruta a la vez, teniendo así una estrategia de abastecimiento mucho más controlada.

### **Operador 10 – Consolidación estratégica completa**

Este operador es sumamente disruptivo y a su vez robusto, ya que su utilización representa una reorganización total de la estrategia logística para un periodo evaluado. Es decir, funciona como un reinicio parcial que permite saltos importantes dentro del espacio de soluciones. La reconstrucción efectuada se divide en dos fases, primero, el algoritmo identifica el CD estratégico dentro del sistema y calcula la demanda total de los clientes en el periodo a modificar. Una vez hecho esto, se reemplazan todas las rutas  $k$  generadas, y se crea un nuevo conjunto de rutas  $k$  que busca enviar exclusivamente al CD estratégico. Seguidamente, ya con el inventario distribuido únicamente en el CD de interés, el algoritmo elimina todas las rutas  $m$  previamente creadas y utiliza la heurística *First Fit Decreasing* definida anteriormente para crear un conjunto de rutas  $m$  nuevo que permita abastecer las necesidades de los clientes. La siguiente figura ejemplifica este operador, asumiendo a CD1 como el CD estratégico.

## Figura

### *Modificación de solución vecina – Consolidación estratégica total*



## Operador 11 – Algoritmo Savings (Fusión por ahorro)

El algoritmo *Savings* planteado para el operador 11 es una versión más robusta y completa del operador 6. Mientras que en el operador 6 se fusionan dos rutas al azar, en este la idea es ejecutar la fusión más rentable posible, generando un ahorro en los costos de transporte. Para su ejecución, el algoritmo elige un CD candidato. Posterior a esto, el algoritmo hace un cálculo de ahorros iterando sobre todos los pares posibles de rutas que parten del CD elegido en el paso anterior. Para cada par de rutas ( $M_A$  y  $M_B$ ) calcula el ahorro que generaría unirlos en una sola ruta. Para la fórmula de ahorro se utiliza la fórmula propuesta por Clarke & Wright, que representa la disminución en distancia (o en este caso costo) al no tener que ir dos veces al depósito. La fórmula se define de la siguiente manera:

$$\text{Ahorro} = [\text{Costo}(\text{fin}_A, \text{CD}) + \text{Costo}(\text{CD}, \text{inicio}_B)] - \text{Costo}(\text{fin}_A, \text{inicio}_B)$$

En donde, “fin\_A” representa al último cliente de la ruta A e “inicio\_B” es el primer cliente de la ruta B. Después de esto, el algoritmo identifica el par de rutas que produce el máximo ahorro, y si este valor es positivo y la carga combinada no excede la capacidad del vehículo, se ejecuta la fusión. La ruta con el menor ahorro se anexa a la de mayor ahorro, y la segunda en la lista queda vacía.