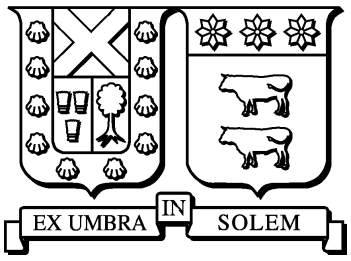
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE INFORMÁTICA

SANTIAGO – CHILE



"UN ACERCAMIENTO METAHEURÍSTICO BASADO EN BÚSQUEDA INFACTIBLE PARA EL PROBLEMA DE RECOLECCIÓN DE LECHE CON SELECCIÓN Y MEZCLA"

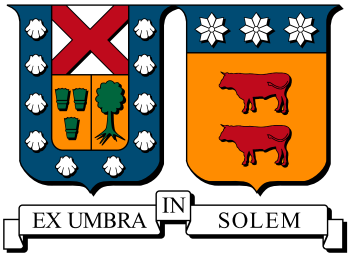
DANIEL IGNACIO ROMÁN VALLEJO

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INFORMÁTICO

PROFESOR GUÍA: ELIZABETH MONTERO U.

MARZO 2024

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO – CHILE



"UN ACERCAMIENTO METAHEURÍSTICO BASADO EN BÚSQUEDA INFACTIBLE PARA EL PROBLEMA DE RECOLECCIÓN DE LECHE CON SELECCIÓN Y MEZCLA"

DANIEL IGNACIO ROMÁN VALLEJO

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL INFORMÁTICO

PROFESOR GUÍA: ELIZABETH MONTERO U.
PROFESOR CORREFERENTE: JOSÉ LUIS MARTÍ L.

MARZO 2024

Agradecimientos

Agradezco a mi familia, tanto a mis padres como a mi hermano Nicolás, que también vivirá este proceso pronto en la universidad y espero poder ayudar como el me ayudo a mí.

Agradezco a mis amigos, compañeros de universidad quienes siempre estuvieron cuando necesité ayuda en el proceso completo vivido en la universidad.

Agradezco a los profesores de la universidad, que siempre tuvieron buena disposición y me parecían personas admirables; el profesor Nicolás Rojas, que me ayudó cuando estuve en una mala situación personal, a la profesora Erika Rosas que siempre consideré muy buena enseñando y volvió a despertar mi gusto por la teoría y la utilidad de los conocimientos sobre redes. También al profesor y jefe de carrera José Luis Martí que siempre me sorprendió lo buen profesor que era, su buena memoria, su seguridad, y su capacidad para hacer muchas cosas a la vez.

Finalmente agradezco a la profesora Elizabeth Montero, quien me apoyó bastante en todo el proceso, muy atenta desde el primer momento que empecé a tener complicaciones mientras trabajaba en este escrito. Además que siempre disfrutaba de su alegría y ánimos para trabajar, o incluso para hablar sobre temas variados o de contingencia.

Resumen

En esta memoria se aborda el problema de Recolección de Leche con Selección y Mezcla utilizando técnicas meta-heurísticas.

El problema consta de una flota de camiones que poseen cierta capacidad en términos de volumen, granjas en distintas locaciones geográficas, que producen leche de distintas calidades, y una planta de procesamiento que demanda una cierta cantidad de esta leche. Además la leche que se recolecta puede ser mezclada tanto al momento de ser recogida por el camión, como en la planta de procesamiento, haciendo al problema de seleccionar las granjas que cada camión visitará, y en qué orden, con el fin de maximizar los beneficios obtenidos de la leche y reducir los gastos relacionados al transporte.

En esta memoria se propone una técnica meta-heurística que trabaja sobre el espacio infactible de tipo búsqueda local iterativa (Iterated Local Search (ILS)), con una fase de intensificación y una de exploración, de tal manera de obtener rutas que logren generar mayores beneficios, mientras se suple la demanda mínima de la planta.

Los resultados obtenidos de la ejecución de este método presenta un rendimiento que no supera a los que existen previamente a la literatura utilizada de referencia, pero muestra resultados prometedores cuando es utilizado en conjuntos grandes de granjas, los cuales llegan a ser hasta un 20.6 % mejores que los obtenidos por otros autores.

Índice de Contenidos

Índice de tablas

Índice de figuras

Glosario

MCPSB: Problema de recolección de leche con selección y mezclas (Milk Collection Problem with Selection and Blending).

SA: Recocido simulado (Simulated Annealing).

HC: Hill Climbing.

HCFI: Hill Climbing First Improvement.

ILS: Iterative Local Search

MPVRP: Multiple Product Vehicle Routing Problem.

PCP: Prize Collecting Problem.

PCVRP: Prize Collectng Vehicle Routing Problem.

TSP: Traveling Salesman Problem.

VRP: Vehicle Routing Problem.

Introducción

En el sur de Chile se concentra la mayor parte de la industria productora de leche del país. Debido a esto, las plantas procesadoras de lácteos también se asentaron cerca, por lo que compran y recolectan leche para satisfacer la cantidad que necesitan para la fabricación de productos como lo son los distintos tipos de quesos, el yogur y la mantequilla. Las plantas ven reflejado en el precio de sus productos el gasto producido de comprar la leche, transportarla y los procesos para tratarlas hasta llegar al producto final. De acuerdo a la *Food and Agriculture Organization* (FAO) los costos del transporte de leche corresponde a más de un 30 % del costo total de esta [10].

Estas plantas procesadoras de leche disponen de flotas de camiones de transporte, que se encargan de recorrer distintas rutas que recolectan leche de granjas de producción, para posteriormente volver a depositarla en la planta que las procesará. Esto involucra un desafío importante porque muchas veces se tiene una demanda de leche que implica visitar cientos de granjas de producción. Dado lo anterior, es que surge el Milk Collection Problem (MCP), el cual es un problema de optimización combinatoria que busca maximizar los beneficios obtenidos al recolectar leche. En este trabajo se considera una variante que considera la mezcla de leche, tanto en el camión al momento de visitar una granja, como al momento de depositar esta en la planta procesadora, lo que da origen al Milk Collection Problem with Selection and Blending (MCPSB).

Este trabajo propone un método de búsqueda incompleta para resolver MCPSB, utilizando como referencia una instancia real del problema ubicada en el sur de Chile, permitiendo mezclas entre tres categorías de leche. Además se considera en este trabajo el permitir generar soluciones no factibles, es decir que permita generar soluciones que no cumplan la demanda

mínima requerida por la planta de procesamiento.

Este trabajo define un acercamiento meta-heurístico basado en Iterated local search (ILS), que permite acercarse a un óptimo local dentro de un espacio de posibles soluciones.

Este documento se organiza de la siguiente manera. En el primer capítulo se define el problema, se introduce el contexto real en más profundidad, junto a un ejemplo del funcionamiento de las mezclas, también se presenta la formulación matemática de este, explicando en detalle todos los parámetros, variables y restricciones que posee. En el capítulo 2, se presenta el estado del arte del problema de enrutamiento de vehículos (VRP), y distintas variantes que involucren recolección de elementos de distinto tipo, múltiples productos, también así como otros problemas de recolección de leche donde se utilizan camiones con múltiples compartimientos, y otras variantes del problema de recolección de leche en sí. En el tercer capítulo se presenta el método de resolución propuesto, se explica la representación de las soluciones, la función de calidad, la estructura general de resolución y cada componente que es parte del proceso de obtención de soluciones. En el capítulo cuatro se presenta el conjunto de datos que se utiliza evaluar las distintas soluciones, se presentan tanto instancias de pruebas sintéticas como una instancia real del problema. Además se incluye la sintonización de parámetros realizad. Los resultados fueron comparados con métodos de la literatura que ocuparon el mismo conjunto de instancias. Finalmente, en el quinto capítulo se presentan las conclusiones generales del trabajo realizado en esta memoria, y se proponen posibles mejoras y lineamientos que puedan mejorar este método propuesto.

El objetivo general de este trabajo de memoria es diseñar una técnica metaheurística basada en búsqueda infactible para resolver el problema de recolección de leche con selección y mezcla.

Los objetivos específicos son:

- Describir las técnicas utilizadas en la literatura para resolver el problema de recolección de leche.
- Diseñar una técnica que utilice búsqueda infactible para resolver el problema de recolección de leche con selección y mezcla.

- Comparar el desempeño de las técnica propuesta con aquellas existentes en la literatura del problema.

Capítulo 1

Definición del Problema

En el sur de Chile se concentra la mayor parte de la industria productora de leche del país. Debido a esto, las plantas procesadoras de lácteos también se asentaron cerca, por lo que compran y recolectan leche para satisfacer la cantidad que necesitan para la fabricación de productos como lo son los distintos tipos de quesos, el yogur y la mantequilla. Por lo que estas plantas ven reflejado en el precio de sus productos el gasto producido de comprar la leche, transportarla y los procesos para tratarlas hasta llegar al producto final. De acuerdo a la *Food and Agriculture Organization* (FAO) los costos del transporte de leche corresponde a más de un 30 % del costo total de esta [10].

Un problema de recolección de leche con selección y mezcla, también denominado con sus siglas en inglés MCPSB (*Milk Collection Problem with Selection and Blending*), consiste en un problema de optimización combinatoria, el cual se basa en la recolección de leche de distintas calidades en diferentes predios de producción. El problema se centra en el proceso de recolección de leche desde distintas granjas distribuidas geográficamente en una amplia zona. Los camiones que se encargan de transportar la leche poseen un único contenedor, lo que implica que si se recolecta desde granjas que producen distintas calidades de leche esta se mezclará en el contenedor. Por otro lado, el problema considera que no todas las granjas deben ser visitadas, pero establecer una demanda mínima de cada calidad de leche. El objetivo del problema es maximizar las ganancias asociadas a la leche recolectada menos el costo de transporte del proceso. Esto obliga a generar rutas que por una parte aseguren el

cumplimiento de las cuotas especificadas, pero a la vez aprovechen la mezcla en contenedor para reducir costos de transporte asociados a la estructura de las rutas. En el caso que no se cumpla con la demanda de una de estas calidades es posible realizar mezcla en la misma planta con los excesos de las otras.

El objetivo de esto es generar una ruta óptima para cada camión de transporte, en la cual se indiquen las granjas a recorrer y su respectivo itinerario, buscando así una maximización de las utilidades de la planta de procesamiento, con la utilización de viajes cortos en los cuales la leche transportada permita mezclar las calidades de la misma, mientras se suple la demanda.

El problema de recolección de leche con selección y mezcla, es una de las múltiples variantes que existen del VRP, por lo que este también se considera un problema *NP-difícil* [11]. El problema estudiado en este trabajo de memoria se compone de un conjunto de camiones homogéneos, es decir, tienen la misma capacidad de transporte, un conjunto de calidades de leche y finalmente un conjunto con todos los nodos, los cuales incluyen a los predios de producción de leche y a la planta de procesamiento. Esta última es la que requiere satisfacer una demanda mínima de cada una de las distintas calidades de leche, permitiendo el sobre abastecimiento de un tipo de éstas, cuando pudiese generar un beneficio mayor.

El conjunto de nodos se ubica en puntos geográficos determinados.La figura 1.1 muestra en un mapa la disposición de 40 granjas de producción. Estas ubicaciones se utilizan permiten calcular las distancias entre cada par de nodos especificando el problema como un grafo totalmente conexo. Cada uno de éstos produce leche de una calidad dada, y el camión que visite una granja deber recolectar la totalidad de la leche producida en esta, lo que implica que cada nodo es visitado solo una vez. Además se requiere suplir la demanda de cada tipo de leche, por lo que no es necesario visitar todas las granjas, solo las suficientes para suplir la necesidad de la planta de procesamiento, esto implica que se tiene un problema de generación de rutas con selección de nodos.

De acuerdo a la legislación chilena vigente [2], la leche que se produce en las distintas granjas se clasifica en tres tipos de calidades. Dichas calidades especifican la posibilidad de utilizar dicha leche en la elaboración de distintos productos lácteos. El nivel A identifica la

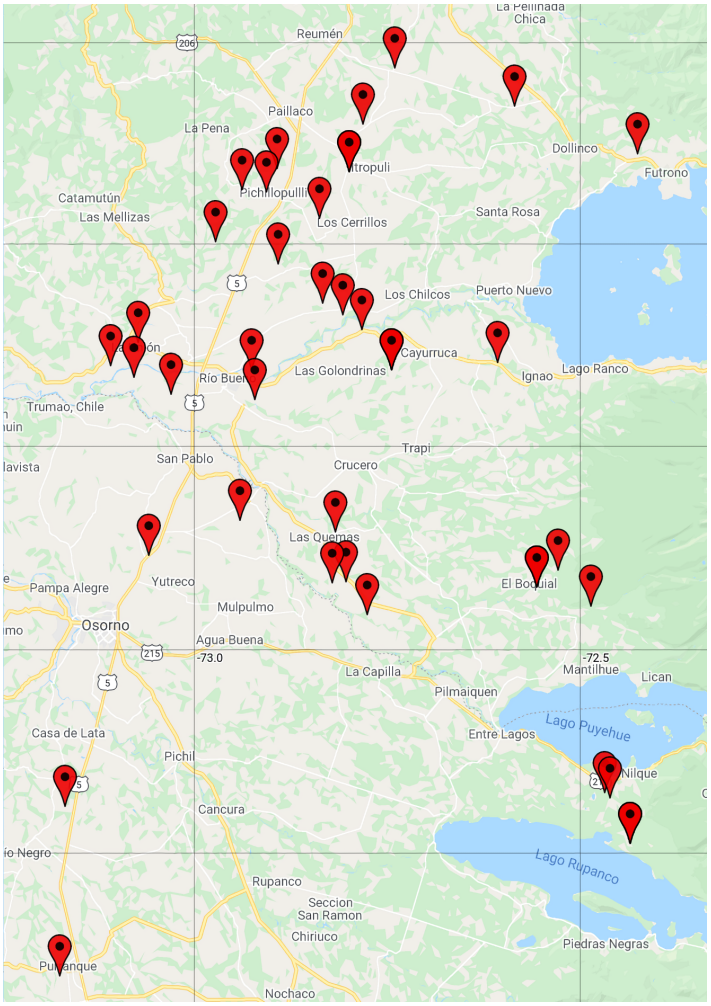
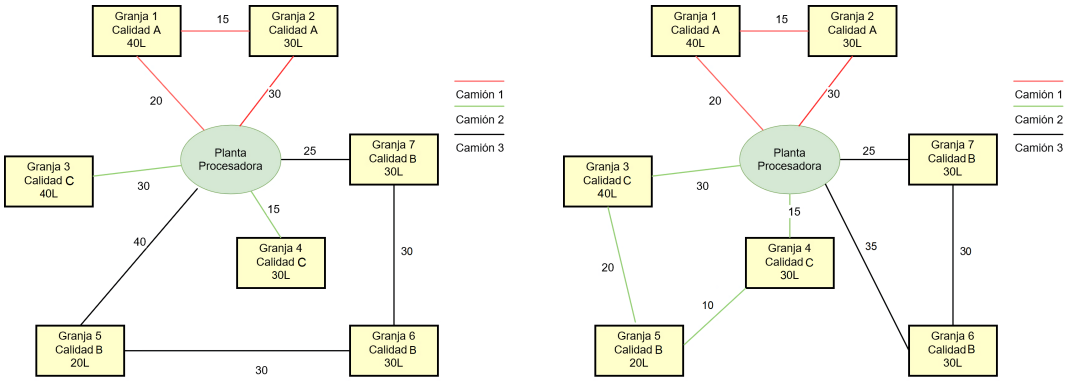


Figura 1.1: Disposición de 40 granjas productoras de leche en el sur de Chile.
Fuente: Elaboración Propia

leche de mejor calidad, el nivel B especifica leche de calidad intermedia y el nivel C identifica la leche del nivel más bajo. Si se tienen camiones que solo poseen un compartimiento, solo es posible almacenar leche de una sola calidad, lo que restringiría muchas rutas posibles, por lo que se considera la posibilidad de mezclar las leches. Se define así un conjunto de reglas para combinar dos tipos de calidades diferentes. Así, se considera que la mezcla de dos producciones de diferente calidad resulta en la suma de los litros, pero categorizada como la de peor calidad. Por ejemplo, si se mezcla leche de calidad A con una de calidad B se considera que la mezcla de dichas producciones será considerada de calidad B. Debido a lo anterior, la cantidad de rutas posibles aumenta, respecto al caso que no considera mezclas. La figura 1.2 muestra un ejemplo en el que hay que satisfacer una demanda de 60 litros de

cada tipo de leche, en ambos casos expuestos en la figura se suple la demanda, aunque en el segundo se utiliza mezcla en ruta de dos calidades en el camión 2, haciendo que 20L de leche de calidad B se sume a la de calidad C. Gracias a lo anterior, se logra reducir la distancia total recorrida de 235 a 230 unidades de distancia.



(a) Sin Mezcla (b) Con Mezcla

Figura 1.2: Posible asignación de rutas con y sin mezcla.
Fuente: Elaboración Propia

También se considera que los camiones inician y terminan su recorrido en la planta procesadora y se asume que solo pueden realizar una ruta utilizando la flota completa de camiones.

En la siguiente sección se presenta la formulación matemática del problema de recolección de leche con selección y mezcla estudiado en este trabajo de memoria.

1.1. Formulación Matemática

El modelo matemático que se presenta en esta sección corresponde al modelo presentado en [16]. Primero se definen los parámetros asociados al problema presentado. Luego se presentan las variables de decisión utilizadas en la formulación. Después se define una función objetivo para medir la calidad de las soluciones obtenidas. Finalmente se describen todas las restricciones a las que está sujeto el problema presentado en este trabajo de memoria.

1.1.1. Parámetros

Se definen los siguientes componentes que se utilizaran en ecuaciones posteriores.

- $G(N, A)$, siendo G un grafo completo.
- N, Conjunto de nodos, incluyendo tanto a los productores de leche como a la planta de procesamiento.
- A, Conjunto de arcos entre los nodos productores de leche y la planta de procesamiento.
- n_0, El nodo que corresponde a la planta de procesamiento.
- A^0, Conjunto de arcos entre la planta procesadora y cada una de las granjas.
- K, Conjunto de camiones de recolección de leche.
- T, Conjunto de calidades de leche.
- N^t, Conjunto de granjas que producen leche de calidad t, $\forall t \in T$
- D^t, Calidad de leche resultante, mezcla de r y t resulta en t, $\forall t$, $r \in T$
- IT, Conjunto de pares ordenados (i, t),$\forall i \in N$,$\forall t \in T$ dado que cada granja produce leche de una única calidad.
- Q^k, Capacidad del camión k,$\forall k \in K$. Considerando que todos los camiones poseen la misma capacidad.
- q_i^t, Cantidad de leche de calidad t producida por la granja i.
- d_{ij}, Distancia del arco $(i, j) \in A \cup A^0$.
- L^t, Precio de venta de la leche de calidad t, $\forall t \in T$.
- C, Costo de transporte por kilómetro recorrido.
- p^t Demanda de leche de calidad t, $\forall t \in T$.

1.1.2. Variables

Se definen las siguiente variables de decisión para el problema:

- $x_{ij}^{k} = \begin{cases} 1, \text{ si el camión viaja del nodo } i \text{ al } j. \ \forall k \in K, \forall (i,j) \in A \\ 0, \text{ si no} \end{cases}$
- $y_{i}^{kt} = \begin{cases} 1, \text{ si el camión recolecta leche de calidad } t \text{ en la granja } i. \ \forall k \in K, \forall t \in T, \forall i \in N \\ 0, \text{ si no} \end{cases}$
- $z^{kt} = \begin{cases} 1, \text{ si el camión k entrega leche de calidad t a la planta. } \forall k \in K, \forall t \in T \\ 0, \text{ si no} \end{cases}$
- w^{kt}: Volumen de leche de calidad t que es entregado en la planta por el camión k, $\forall k \in K, \forall t \in T$.
- v^{tr}: Volumen de leche de calidad t entregado a la planta, que se utilizará como leche de calidad r,$\forall r, t \in T$.

Entonces el dominio de las variables corresponde a:

$$x_{ij}^{k}, y_{i}^{kt}, z^{kt} \in \{0,1\} \qquad \forall i \in N, (i,j) \in A, k \in K, t \in T$$

$$w^{kt}, v^{tr} \geq 0 \qquad \forall k \in K, t, r \in T, r \in D^{t}$$

1.1.3. Función objetivo

El objetivo de este problema es encontrar rutas que maximizan los beneficios obtenidos por la recolección de leche de distintas calidades, y su función objetivo corresponde a:

$$fo = Max \sum_{t \in T} \sum_{r \in T} L^{r} v^{tr} \ - \ C \sum_{k \in K} \sum_{(i,j) \in F} X_{ij}^{k} d_{ij} \tag{1.1}$$

Es decir, la función objetivo es la diferencia entre las ganancias obtenidas de la leche recolectada y mezclada, y los costos de transportarla.

1.1.4. Restricciones

1. La cantidad de leche transportada por un camión no puede superar la capacidad máxima del mismo.

$$\sum_{t \in T} \sum_{i \in N:(i,t) \in IT} q_i^t y_i^{kt} \leq Q^k \qquad \forall k \in K$$

2. Cada granja puede ser visitada a lo más por un camión.

$$\sum_{k \in K} y_i^{kt} \leq 1 \qquad \forall i \in N, t \in T : (i,t) \in IT$$

3. Todos los camiones deben tener una ruta, y debe iniciar en la planta.

$$\sum_{j \in N:(0,j) \in A} x_{0j}^k = 1 \qquad \forall k \in K$$

4. Recorrido continuo de los camiones, no se permiten saltos.

$$\sum_{j \in N_0:(i,j) \in A} x_{ij}^k = \sum_{h \in N_0:(j,h) \in A} x_{jh}^k \qquad \forall k \in K, j \in N_0$$

5. Si el camión recolecta leche en la granja i, debe pasar por ella.

$$\sum_{h \in N_0:(h,i) \in A} x_{hi}^k = y_i^{kt} \qquad \forall i \in N, k \in K, t \in T : (i,t) \in IT$$

6. Se debe suplir la demanda de cada tipo de leche.

$$\sum_{t \in T} v^{tr} \geq P^r \qquad \forall r \in D^t$$

7. Toda la leche recolectada debe ser entregada en la planta.

$$\sum_{k \in K} \sum_{t \in T} w^{kt} = \sum_{(i,t) \in IT} q_i^t$$

8. Balance de la cantidad de leche de cada calidad que llega a la planta, y la cantidad resultante de la mezcla en la planta.

$$\sum_{r \in D^t} v^{tr} = \sum_{k \in K} w^{kt} \qquad \forall t \in T$$

9. Evita que un camión cargue una leche de una calidad menor que t, si es que va a entregar una de calidad t a la planta.

$$z^{kt} \leq 1 - \sum_{r \in D^t : r \neq t, (i,r) \in IT} y_i^{kr} \qquad \forall k \in K, i \in N, t \in T$$

10. La leche que entrega una camión a la planta es de una sola calidad, pero puede estar mezclada.

$$\sum_{t \in T} z^{kt} \leq 1 \qquad \forall k \in K$$

1.2. Resumen

En este capítulo se contextualiza y describe en detalle el problema de recolección de leche con selección y mezclas que se abordará en este trabajo de memoria. Este problema consiste en maximizar el beneficio de una planta de procesamiento de productos lácteos. La planta trabaja con leche de tres calidades, la cual es recolectada por camiones homogéneos que poseen un único compartimento para la leche. Debido a esto se considera la posibilidad de mezclar leche de distintas calidades en un solo camión, pero esto implica una perdida de calidad, lo que a su vez puede generar rutas más eficientes en términos de costos de transporte. Por otro lado, se considera una demanda de cada tipo de leche por parte de la planta procesadora, por lo cual no es necesario recolectar la leche de todas las granjas. Por

otro lado, también es posible realizar mezclas de leche en la misma planta. Finalmente se considera como función objetivo la maximización de la ganancia generada por la utilidad debida a la leche recolectada y la minimización de los costos asociados al transporte de ésta.

Capítulo 2

Estado del Arte

El *Milk Collection Problem with Selection and Blending* (MCPSB) puede definirse como una combinación de dos problemas clásicos de optimización combinatoria, siendo uno de éstos la generación de rutas de vehículos, conocido como VRP (*Vehicle Routing Problem*) [6], y el problema de la mochila (*Knapsack Problem*) [7]. En el primero se busca generar un conjunto óptimo de rutas para satisfacer la demanda de un grupo de clientes, y el segundo busca el conjunto óptimo de objetos a llevar en una mochila sin sobrepasar la capacidad de ésta. Dado esto, se puede considerar que el problema de la mochila es equivalente al de seleccionar las granjas a visitar, y el ruteo de vehículos es el que decide qué vehículo visitará a cda cliente y en qué orden.

En este capítulo se contextualizará en cuanto a la literatura relacionada con el problema de ruteo de vehículos, algunas de sus variantes, y las diferencias de éstas con el problema estudiado. Se presentará el problema de ruteo de vehículos clásico, algunas de sus principales variantes y los estudios que consideran la recolección de leche como una variante de este.

2.1. Problema de ruteo de vehículos

Dantiz y Ramser en 1959 [6] son los primeros en estudiar el ruteo de vehículos en un problema de distribución de combustible. En este se estudio se logra concluir que el problema

es *NP-difícil*, debido a que se puede considerar una generalización del problema del vendedor viajero (TSP) [9]. Como en el problema del vendedor viajero se busca generar la ruta con distancias más cortas para visitar todos los nodos, en los problema de ruteo de vehículos el desafío es esencialmente el mismo, pero con la diferencia que se pueden realizar múltiples circuitos para visitar todas las ciudades.

En [6] se presenta un modelo matemático de programación lineal entera para el problema. Además, se hace un análisis sobre variantes posibles a las determinadas en las restricciones, considerando por ejemplo el caso en el que se tiene una demanda de múltiples productos líquidos. También se define el caso de que los camiones tengan distintas capacidades. Finalmente se estudia el caso en el que la demanda de cada estación de combustible sea satisfecha tanto por una sola entrega como por múltiples entregas de menor cantidad, lo que puede reducir las distancias recorridas por el grupo de vehículos completo.

A continuación se presenta una revisión de acercamientos asociados a variantes de interés del problema de ruteo de vehículos como lo son el problema de ruteo de vehículos con recolección de premios, con recolección de múltiples productos, diferenciando dos variables para este último.

2.2. Problema de ruteo de vehículos con recolección de premios - PCVRP

Una de las variantes del problema de ruteo de vehículos es el problema de ruteo con recolección de premios, el cuál consiste en aplicar a la recolección de bienes un Problema de Recolección de Premios (PCP). Este problema es presentado por Balas en 1989 [3] y es considerado una variante del TSP, el cual consiste en un vendedor que viaja por un conjunto de ciudades, por lo tanto se considera un costo por el viaje realizado a cada una, pero también se genera un beneficio en cada ciudad que es visitada. Este problema tiene como objetivo maximizar el beneficio generado por las ventas mientras se minimiza el costo que involucra el transporte. Dado lo anterior, si se aplica el PCP a un VRP se tiene un PCVRP, y un ejemplo de este sería el MCP, ya que se debe recolectar leche de las granjas, la cual genera

un beneficio, aunque también se genera un costo de transporte de esta.

En [14], Montero et al. resuelven el problema de recolección de leche como un problema de ruteo de vehículos con recolección de premios (PCVRP), aunque a diferencia del planteado en este trabajo de memoria las granjas producen leche de solo una calidad, por lo tanto no se consideran demandas múltiples ni mezclas.

En su trabajo los autores proponen una técnica de búsqueda local para resolver instancias de gran tamaño. Proponiendo un modelo de programación entera y también una metaheurística GRASP [17] que consta de tres fases: preprocesamiento, construcción y postprocesado. Para la construcción de soluciones iniciales se utiliza un algoritmo seudo codicioso en el cuál se determina el orden de visita utilizando dos criterios: demanda pendiente y demanda pendiente respecto a la distancia. La demanda pendiente corresponde a la diferencia entre la demanda insatisfecha de la planta y la producción de las granjas, lo que favorece la selección de las granjas que más leche del tipo pendiente producen. El segundo criterio corresponde al producto entre la demanda pendiente y la distancia a los nodos. Por otro lado en el postprocesado se utilizan cuatro movimientos para el proceso de búsqueda local: eliminación de nodo; remueve un nodo buscando una mejor mejora, intercambio interno; realiza un cambio en el orden de los nodos a visitar solo si mejora la solución, intercambio externo; se cambia un nodo de la ruta por otro que no se ha visitado y mejora la solución, y finalmente adición de nodo; añade un nodo a la ruta de un camión que lo permita mientras se busca el menor deterioro de la solución.

Para evaluar la propuesta, los autores presentan experimentos usando una instancia pequeña que consta de 40 nodos productores, los cuales son un subconjunto de nodos de una planta de procesamiento del sur de Chile. En este caso, se considera que todos los nodos producen leche de una única calidad, Se tiene una instancia base con tres camiones y una demanda de 35.000L, a partir de la cual los autores diseñaron otros siete casos de estudios en los cuales se varía la cantidad de camiones (de tres a siete), su capacidad (14.000L a 30.000L), y la demanda (de 35.000L a 100.000L). Los resultados que se obtuvieron con el modelo de programación lineal entera, utilizando el solver CPLEX 12.7, mostraron una mejoría del 33 % para el caso base de 40 nodos respecto a la asignación manual actual de la planta. La solución propuesta por el algoritmo GRASP reduce el costo de transporte, pero

para los otros siete casos se va notando que a medida que el tamaño del problema crece, va aumentando también la brecha entre la solución encontrada y una solución factible encontrada por el modelo lineal. Por otro lado con el acercamiento GRASP, primero se evalúa el impacto de cada movimiento (demanda pendiente, demanda pendiente respecto a la distancia ,eliminación de nodo, intercambio interno, intercambio externo y adición de nodo) utilizada en las etapas de construcción y postprocesado. A partir de esto se concluye que utilizar todos los movimientos en conjunto genera soluciones finales de mejor calidad. Finalmente se comparan los resultados del caso base original junto a los siete propuestos y se concluye que para instancias más pequeñas, refiriéndose a aquellos que de menor demanda y generan a los más cuatro rutas, es mejor resolverlas con el modelo de programación lineal entera. Por otro lado, para los casos más grandes, de más de cuatro rutas, GRASP generaba soluciones entre un 2 % y hasta casi un 11 % mejores, y estas son generadas en un tercio del tiempo que con el *solver*, por lo que además requiere de menor capacidad computacional.

2.2.1. Problema de ruteo de vehículos multi-producto - MPVRP

El problemas de ruteo de vehículo multi-producto (MPVRP) fue propuesto por Abkowitz y Cheng en 1988 [1], quienes plantean un modelo para transportar sustancias peligrosas, en el cual definen que los camiones solo pueden transportar un tipo de sustancia dado el posible riesgo de una reacción o un accidente. En su trabajo, los autores buscaban minimizar tanto el riesgo como los costos de transportar las sustancias. El MPVRP es otra variante que acerca los problemas de enrutamiento de vehículos al problema de recolección de leche con selección y mezcla, ya que se pueden considerar que las leches de distintas calidades corresponden a distintos productos a transportar.

Los problemas de ruteo de vehículo multi-producto pueden tener distintos acercamientos según las restricciones que se consideren, como son el uso de camiones que pueden transportar solo un tipo de producto, como así también los que tienen la capacidad de transportar múltiples tipos de estos gracias a la utilización de camiones con múltiples compartimientos, además existe la posibilidad de mezclar los productos en estos. A continuación se presentarán algunos trabajos que consideran camiones de un solo compartimiento.

Dooley [8] en 2005 estudia un problema de ruteo de vehículos multi-producto de recolección de leche en Nueva Zelanda. En este trabajo estudian el impacto de recolectar dos tipos de leche en dos grupos de treinta granjas cada uno, donde uno se caracteriza por tener nodos de baja producción pero muy cercanos entre sí, y el otro de nodos que tienen mayor producción pero separados a una mayor distancia. Para esto se consideran camiones que solo poseen un compartimiento, por lo que solo pueden transportar un tipo de leche. A diferencia del problema planteado en esta memoria no se tiene una demanda fija, sino que se debe recolectar toda la leche disponible.

Se generaron seis rutas iniciales utilizando una heurística *greedy* para ser utilizadas posteriormente por el *software* de algoritmo genético "Evolver" [4] con el cual se decidió el orden de cada ruta a recorrer para cada instancia y se compararon los costos de transporte de cada una principalmente. Los autores utilizan un paquete de excel y utilizan como parámetros de tasa de mutación 0.06, tasa de cruzamiento de 0.5, y una población de 1000 individuos.

Para este estudio se generaron cuatro escenarios en los cuales cambiaba el tipo de leche producida en los dos grupos de granjas (30 cada uno), de calidad A a B, en 20 años de producción. El primer escenario consta de no realizar cambios, el segundo considera cambiar la calidad de producción de manera aleatoria a un 25 % de las granjas. El tercer escenario cambia la calidad de producción de manera aleatoria a un 50 % de las granjas. El último cambia la totalidad de las producciones de las granjas.

A partir de este trabajo se concluye que los costos de recolectar leche no son altos considerando el beneficio generado, además que los costos al cambiar el tipo de producción de las granjas aumentaba de un 4.5 % a un 22.0 %, esto último considerando que entre menos granjas cambiaban, mayor era la diferencia de los costes. Esto debido a que resultaba menos costoso transportar leche de calidad de tipo B en comparación a la tipo A, por lo que a medida que aumentaba la cantidad de tipo B se reducían los costes.

2.3. Problema de recolección de leche con selección y mezclas - MCPSB

El problema de recolección de leche con selección y mezclas (MCPSB) correspone una combinación de problemas clásicos de enrutamiento de vehículos, y es la variante en la que se trabajará en este trabajo de memoria.

En 2020 Villagrán et. al. [20] vuelven a trabajar en un problema de recolección de leche, pero esta vez consideran una versión del problema en que las leches poseen diferentes calidades, lo que permite considerarlo un problema de ruteo de vehículos multi-producto con mezcla.

En su trabajo los autores proponen un algoritmo de búsqueda local iterada para resolver instancias de gran tamaño, que no pueden ser resueltas con algoritmos de búsqueda completa. Para esto primero se inició de una solución aleatoria, que puede no ser factible, para trabajar con soluciones no factibles se penalizan los litros de demanda no satisfecha y cuando el camión excede su capacidad máxima de transporte. Luego de tener una solución inicial se busca en el vecindario de esta alguna de mejor calidad. Así, los autores realizan dos movimientos: el primera busca generar nuevas rutas realizando un intercambio externo de un nodo aleatorio entre estas, el segundo consta de buscar una mejor calidad realizando un intercambio interno del orden de visita entre dos nodos de la ruta (2-*opt*). Además el algoritmo que realiza la búsqueda local utiliza una variable *global_local* para detectar el estancamiento de haber realizado alguno de los dos movimientos, aunque también se tienen variables *extra_local* e *intra_local* para almacenar la mejor solución obtenida al realizar una búsqueda local con cada uno de los movimientos.

Después de esto se reemplaza la solución con la mejor obtenida de estos procedimientos. En caso de detectar la condición de estancamiento, se realiza una perturbación a la mejor solución actual. La perturbación consiste en poner una granja aleatoria en cualquier ruta y en cualquier posición de ésta, lo que puede resultar en un cambio interno o externo de ésta e incluso puede generar una solución no factible. De lo anterior cabe destacar que existe un factor de perturbación que controla cuántas de dichas inserciones se realizan.

Para evaluar este método se utilizaron dos tipos de instancias, la primera corresponde a 37 instancias conocidas de VRP que fueron adaptadas a un MCP, instancias consideradas como pequeñas y que constan de tres camiones. Para el segundo tipo se tiene una instancia real que consta de 500 granjas productoras de leche del sur de Chile y 100 camiones. Además de esta última se generaron tres sub-instancias que constan de la misma cantidad de granjas pero reduce la cantidad de camiones a 88, 77 y 72, debido a que no se consideran los que tienen menor capacidad.

Primero se analiza el efecto del factor de perturbación, para ello se proponen cuatro valores para éste (0,05, 0,10, 0,15 y 0,20) y se comparan tanto en calidad como en tiempo con los obtenido por Paredes et al. [16]. A partir de esto se pudo notar que, en general, lo que sucede es que en la mayoría de los casos a medida que el factor de perturbación aumenta, también lo hace el tiempo que tarda en encontrar la solución óptima encontrada. Considerando el caso real inicial, se comparó el método propuesto con *hill-climbing* y lograron demostrar que el *iterated local search* propuesto llegaba a mejores resultados, dado que *hill-climbing* ni siquiera logró alcanzar a los propuestos por Paredes. Posteriormente se compara el rendimiento del algoritmo de búsqueda propuesta entre las cuatro instancias del caso real, y logran notar que todos generan resultados similares a excepción de la instancia con 77 camiones, la cual obtenía mejores resultados con mayor frecuencia. Luego de esto se realizó un estudio de la convergencia de los resultados a largo plazo, con un límite de tiempo de 26.500 segundos el caso y inicial, y el que utiliza 77 camiones llegan a una solución muy similar, pero que supera a la obtenida en [16] y tardando la mitad de tiempo que esta última.

En [19], la autora resuelve un problema de recolección de leche con selección y mezcla, con tres calidades de leche y con camiones de igual capacidad, basándose en las mismas 500 granjas utilizadas por los autores de [16].

Se propone la utilización de un algoritmo *Simulated Annealing* (SA) [12], que implementa una fase de preprocesamiento, construcción y de postprocesado. Para las soluciones iniciales se utiliza un algoritmo *pseudo-greedy* con ruleta para seleccionar el siguiente nodo de la ruta. En este caso las rutas se construyen asignando nodos que producen leche de mejor calidad primero hasta suplir la demanda de este tipo, y así sucesivamente con las calidades restantes. Lo anterior se debe a la necesidad de suplir la demanda mínima de la leche de máxima

calidad, dado que al realizar mezclas de diferentes tipos el resultado es considerado como la de menor calidad de la mezcla.

Para seleccionar el siguiente nodo de la ruta el algoritmo *pseudo-greedy* genera una lista candidata de las granjas cuyas producciones no superan la capacidad restante del camión, y son de la misma o mejor calidad de la que está recolectando. Luego de tener la lista candidata, se favorece a los nodos con mayor producción utilizando una ruleta. Finalmente, para la fase de postprocesamiento, se utilizan cuatro movimientos: romper demandas selecciona una ruta aleatoria y de esta elimina al azar un nodo de ella hasta que se deje de suplir la demanda mínima de alguna calidad de leche. El movimiento agregar nodos selecciona una ruta que transporte leche de la mejor calidad y se añade en una posición aleatoria la granja que genere mejor beneficio y que no haya sido visitado. El movimiento reordenar selecciona un nodo de una ruta y se reubica en todas las maneras posibles hasta encontrar el orden que tenga menor costo. Por último, el movimiento remover nodos remueve los nodos que involucran un mayor costo de transporte que el beneficio que generan, pero se debe seguir cumpliendo la demanda.

Los dos primeros utilizan SA, considerando la temperatura del sistema y el beneficio generado al realizar los movimientos. Por otro lado, los últimos dos movimientos, reordenar y remover nodos, buscan mejorar la solución quitando de la ruta los nodos que no generan beneficio y reordenando las rutas para buscar una mejoría en los costos de transporte.

Para evaluar este algoritmo se realizan dos experimentos. El primer experimento considera un conjunto de instancias pequeñas de 40 nodos extraídos desde el conjunto de 500 granjas utilizada por Paredes-Belmar [16]. El segundo considera un conjunto de instancias que constan de agrupaciones del total de 500 granjas. Para las instancias pequeñas se generaron seis casos de estudio, en los cuales se utilizan de cuatro a siete camiones. Dos de estos casos no presentan demanda, lo que hace que el problema se relaje y añade dificultad porque en un inicio cualquier solución generada sería factible. Para el caso real de 500 nodos, se utilizaron seis casos. El primero siendo el caso completo y los otros cinco son agrupaciones realizadas utilizando *k-means* [13], lo que generó instancias que poseen de 31 a 143 nodos.

Para estudiar los resultados obtenidos en las instancias pequeñas, se analizó primero el

cómo afecta cada componente del algoritmo a las soluciones encontradas, y se concluye que los parámetros sintonizados generan mejores resultados en general, aunque ocurre en algunas instancias que valores distintos a estos genera mejores resultados. Posteriormente se comparan los resultado obtenidos con el con los obtenidos utilizando el *solver* CPLEX, de lo que se obtuvieron finalmente soluciones que son de un 0.4 % a un 19.1 % peores a las obtenidas por el *solver*. Para obtener dichos resultados el solver llega a tardar 30 minutos en ejecutarse, mientras que el algoritmo propuesto llega a los 5 minutos de ejecución. Para los casos reales, el algoritmo propuesto obtuvo resultados en un tiempo promedio de 38 minutos, y para el caso completo de 500 nodos tardó 9 horas y 30 minutos, esto implica que no se pueden obtener soluciones en un tiempo razonable con el *solver*, y los distintas instancias no se pueden comparar entre sí, dado que presentan diferencias importantes en algunos casos, dado el *k-means*.

Uno de los estudios mas recientes fue realizado por Moreno [15] en 2021 quien resuelve el mismo problema abordado por Soto y el de este trabajo de memoria, es decir, el problema ruteo de vehículos con selección y mezcla. Además este trabajo se centra en la utilización de meta-heurísticas híbridas, enfocándose en buscar buenos resultados para instancias grandes.

En la memoria de Moreno se trabaja con metaheurísticas híbridas, las cuales se utilizan en distintas versiones de la técnica que se propone. La propuesta se inspira en los algoritmos: *Greedy, Simulated Annealing* y *Hill Climbing First Improvement* (HCFI). Además presenta un diseño de dos etapas que combina SA y HCFI. En primer lugar para construir soluciones iniciales, se utiliza un algoritmo *Greedy* aleatorizado que satisface la demanda mínima de cada calidad de manera iterativa. Esto comenzando desde la calidad más alta a la más baja. Para lograr esto, utiliza una función miope que elige de manera aleatoria uno de los nodos desde un conjunto de los n mejores nodos disponibles. Además se definen operadores de diversificación e intensificación: agregar nodos, remover nodos e intercambios internos y externos en la ruta. El movimiento de agregar nodos consiste en seleccionar una ruta aleatoria y añadir un nodo al azar que no supere la capacidad máxima del camión. El operador eliminar nodos quita de una ruta al azar una granja manteniendo la demanda mínima. Por último, considera también intercambio de nodos interno y externo.

Luego se proponen las variantes basadas en *Greedy* y en SA, teniendo en primer lugar un

Greedy Aleatorizado (G-A), el cual realiza una selección aleatoria del mejor nodo añadido en cada iteración. Además se propone un *Simple Simulated Annealing* (SSA), el que se utiliza después de haber construido una solución inicial factible con G-A, se realizan operaciones que añaden y remueven nodos de manera aleatoria, se utilizan los operadores agregar y eliminar nodos de manera aleatoria, utilizando un *scheduling* de temperatura exponencial que permite aceptar varios movimientos que modifican la solución al inicio de cada ciclo, para que luego esta temperatura disminuya y se realicen movimientos para mejorar la solución actual. Posteriormente se incluye una variante a este algoritmo, denominado *Two Phases Simulated Annealing* (2SA) en el que además de añadir y remover nodos se realizan intercambios internos y externos a las rutas, por lo que es capaz de diversificar e intensificar. Este cambio se utiliza una cantidad fija de iteraciones. Luego se vuelve a trabajar sobre este 2SA, teniendo así la variante *Improved Two Phases Simulated Annealing* (I2SA), la cual mejora las condiciones de cambio entre etapas, utilizando un parámetro para detectar posibles estancamientos, y también otro parámetro que se encarga de determinar cuántas iteraciones se realizan en cada una de las etapas. Finalmente se implementó un retraso en el primer cambio de etapas, dado que se debiese cambiar el espacio de búsqueda luego de obtener una solución inicial, para esto se utiliza un factor de intensificación que permite realizar más etapas de diversificación, y también se cambia el *scheduling* de temperatura exponencial por uno lineal. Finalmente, el último algoritmo propuesto es *Two Phases Routing Algorithm* (2PRA), que busca aprovechar más los recursos en encontrar óptimos locales, por lo que cada vez que se encuentre una nueva región para explotar, se realiza una búsqueda de soluciones vecinas de mayor calidad, cambiando así el SA del algoritmo anterior por uno inspirado en *Hill Climbing* para la etapa de intensificación, pero manteniendo ambos movimientos de intercambio.

Para estudiar la eficacia de estos algoritmos se utilizaron las mismas instancias trabajadas en [19], tanto el caso pequeño de 40 nodos, como el caso real de 500 con los cinco subconjuntos generados con *k-means*, lo que permitirá comparar a posterior el rendimiento de la memoria previamente mencionada.

Antes de realizar las pruebas en las instancias, se realizó una sintonización manual de parámetros con la instancia de prueba y como conjunto de validación las instancias reales, con esto se generaron los primeros resultados para cada algoritmo. Primero el G-A obtiene

soluciones iniciales factibles, pero estas soluciones son pobres en calidad, incluso muchos de estos los costos superan a la ganancia. En segundo lugar se prueba SSA, en el cual se observaron solo valores positivos, y comparando los casos reales con las instancias sin restricción de demanda, tiene resultados mejores, ya que le permite encontrar más movimientos factibles. Luego se probó con 2SA y los resultados son mejores a los obtenidos por SSA en todas las instancias. El siguiente algoritmo a probar fue I2SA, este presenta mejoras para las instancias pequeñas respecto a los dos SA anteriores, pero en las instancias reales se ve más beneficiado en los casos que tienen mayor cantidad de nodos (128, 142 y 500). Finalmente los resultados para el 2PRA, se logran encontrar nuevos máximos en comparación a la técnica anterior, especialmente para el caso real de 500 nodos.

Los resultados previamente revisados utilizaron una sintonización general, por lo que se decide utilizar un sintonizador Evoca [18] con las instancias reales para 2PRA, ya que este llegó a los mejores máximos en el caso más complejo. Se compara el algoritmo MCSA de [19] y 2PRA en las instancias reales, considerando una demanda mínima para cada tipo de leche, obteniendo resultados similares en las instancias más pequeñas, y solo una leve mejoría en una de las cinco instancias, pero en el caso completo de 500 granjas se pudo observar una mejora significativa respecto a MCSA, ya que se obtuvo un resultado 6.7 % mejor.

Por último se analizan las instancias reales, pero sin considerar una demanda mínima de calidad de leche, haciendo que la técnica propuesta por Moreno (2PRA) genere soluciones de 5.4 % a un 29.7 % mejores que MCSA, y en cuanto a tiempo de ejecución, es menor en tres de las seis instancias, siendo la mejora más significativa la instancia completa que es resuelta 13 veces más rápida. Todo esto implica que 2PRA es una técnica que funciona bien para las instancias más grandes, en este caso de 500 nodos, 35 camiones y las tres calidades de leche.

2.4. Resumen

En esta sección se presenta una breve revisión de variantes comunes del problema de ruteo de vehículos, con un énfasis en los problemas de recolección de leche. Se presentó una revisión de trabajos anteriores de autores destacando inicialmente los problema de recolección que solo trabajan con una única calidad de leche donde se concentra la mayor parte de la investigación. Así también como los acercamientos que trabajan con múltiples tipos de leche, pero considerando el transporte utilizado camiones especialmente equipados con compartimentos para realizar el transporte de los diferentes tipos de leche separadamente. Por último, se presentan las variantes que permiten mezclar las leches de distinto tipo en el mismo camión como también las que realizan mezcla de estas en la planta de procesamiento para suplir la demanda de cada tipo.

La mayoría de las soluciones propuestas en la literatura para este problema trabajaban solo sobre el espacio factible de soluciones. Las soluciones planteadas en los trabajos revisados utilizaban técnicas y métodos clásicos para la resolución de problemas de optimización combinatoria, adaptados al problema, con movimientos tanto de construcción como de reparación de las soluciones.

Capítulo 3

Implementación

En este capítulo se presentan la implementación del algoritmo de búsqueda local propuesto para la resolución del problema de recolección de leche con selección y mezcla. Se presentan primero las componentes básicas del acercamientos como lo son la representación de soluciones y la función de evaluación utilizada. A continuación se presenta la estructura general del acercamiento. Se describen los procesos de construcción de soluciones iniciales, el proceso de búsqueda local y el proceso de perturbación.

3.1. Representación de soluciones

Dado que el problema abordado en esta memoria consiste en la generación de rutas para un conjunto de camiones, se utiliza una representación de listas de arcos para cada ruta asociada a cada camión. Considerando que se trata de un problema de ruteo de vehículos, cada ruta comienza y termina su recorrido en la planta de procesamiento. Así, el nodo inicial del primer arco de cada ruta y el nodo final del último arco de cada ruta corresponden a la planta.

La figura 3.1 muestra un ejemplo de representación de la ruta con mezcla asignada al camión 2 del ejemplo que se presenta en la figura 1.2 del capítulo 1. La secuencia indicada comienza su recorrido en la planta procesadora, luego visita a las granjas 4, 5 y 3 para

finalmente retornar a la planta: $0 \rightarrow 4 \rightarrow 5 \rightarrow 3 \rightarrow 0$.

k_2 :	0→4	4→5	5→3	3→0

Figura 3.1: Ejemplo de representación de rutas.

La representación de la solución, por su parte, corresponde a un conjunto de listas de arcos, donde cada una corresponde a la ruta realizada por cada camión. La figura 3.2 muestra la representación de las tres rutas construidas en el ejemplo de la figura 1.2.

3.2. Función de Calidad

Para evaluar la calidad de las soluciones que se construyen se utiliza la función objetivo que se muestra en la ecuación 1.1 de la sección 1.1.3. Además, se permitirá la infactibilidad de las soluciones respecto a la satisfacción de la demanda en la instancia. Dado esto, se modifica la función objetivo, añadiendo una componente que castigue esta infactibilidad en función del porcentaje faltante de la demanda de cada tipo de leche. Además, tendrá mayor castigo la demanda insatisfecha de leche de calidad A, dado que solo se puede suplir dicha demanda con el mismo tipo de leche. Se considera un castigo medio para la insatisfacción de la leche de calidad B, dado que esta puede ser satisfecha deteriorando leche tipo A. Finalmente se considera un menor castigo para la insatisfacción de demanda de leche tipo C, la

k_1 :	0→1	1→2	2→0

k_2 :	0→4	4→5	5→3	3→0

k_3 :	0→7	7→6	6→0

Figura 3.2: Ejemplo de representación de soluciones.

cual puede ser satisfecha deteriorando tanto leche tipo A como leche tipo B.

$$fo = Max \sum_{t \in T} \sum_{r \in T} L^r v^{tr} \; - \; C \sum_{k \in K} \sum_{(i,j) \in F} X_{ij}^k d_{ij} \; - \; \sum_{t \in T} \sum_{r \in T} F^{tr} \tag{3.1}$$

La ecuación 3.1 muestra cómo se añade un termino a la función objetivo encargado de castigar la calidad de la solución obtenida. Este termino se calcula utilizando la ecuación 3.2 que depende a su vez de dos parámetros.

$$F^{tr} = \frac{p^t - v^{tr}}{p^t} \; * P^t L^t \tag{3.2}$$

La ecuación 3.2 calcula el porcentaje de leche faltante, respecto a la demanda del tipo respectivo y lo pondera por la ganancia que generaría L^r, el cual puede tomar los valores presentado en 3.3, y por un factor P. Este factor se encarga de magnificar el castigo a la solución según la cantidad de leche faltante. Se genera así un castigo mayor para la leche no suplida de más calidad, cuando es de tipo A, se reduce este factor para la leche tipo B, y se reduce aún más para la leche de tipo C. Lo anterior se debe al impacto que genera en la función de calidad la cantidad de leche que falta de cada tipo en la solución.

$$L^r = \begin{cases} 30, \text{ si r = A} \\ 21, \text{ si r = B} \\ 9, \text{ si r = C} \end{cases} \tag{3.3}$$

3.3. Estructura general

El trabajo realizado para la resolución del problema de recolección de leche con selección y mezcla consta de tres pasos principales: Construcción, búsqueda local y perturbación.

El pseudocódigo del algoritmo 1 muestra la estructura general del acercamiento de resolución propuesto. Primero se define la holgura de capacidad para todos los camiones, la cuál representa qué tanto se intenta llenar la capacidad de estos con solo un tipo de leche.

Posteriormente se hace un llamado a la función *feasibleSolution* definido en el algoritmo 2, el que se encarga de construir una solución inicial llenando los camiones hasta satisfacer la demanda de cada tipo de leche, priorizando las de mejor calidad requeridas.

En segundo lugar se busca optimizar el orden de los nodos a visitar en las rutas generadas en la solución inicial utilizando el movimiento 2-opt descrito en la sección 3.5.1. Se realiza una búsqueda en vecindario primera mejora como se muestra en el algoritmo 3. Las mejoras se comparan con la mejor solución del proceso, y se actualiza cuando corresponde.

Posterior a esto se ejecuta el movimiento de perturbación *Purify* definido en la sección 3.6.1. Este se encargará de limpiar una ruta de tal manera que deje en esta solo nodos de tipo de leche predominantes y de la mejor calidad. El procedimiento se ejecutará una cantidad de intentos igual a dos veces la cantidad de rutas en la solución actual, aplicándose a una ruta aleatoria en cada uno. Inmediatamente después de realizado todos los intentos, se definirá como mejor solución en caso que la generada por la perturbación supere a la previamente encontrada. Además, se ejecutará el algoritmo *AddCandidates* definido en la sección 3.6.2. Este procedimiento seleccionará una ruta aleatoria de la solución, y eliminará como máximo y de manera aleatoria, dos nodos a recorrer de este y posteriormente insertará nodos de la misma calidad de la ruta de manera aleatoria hasta completar la capacidad del camión, luego se chequea si la solución actual generada mejora a la mejor solución para así actualizarla si corresponde.

Luego de esta serie de movimientos de ejecuta la fase de perturbación. Esta inicia con una nueva ejecución de *AddCandidates* y luego se realizan intercambios de nodos aleatorios en rutas aleatorias de la mejor solución, una cantidad de veces determinadas por un factor de perturbación. Si esta serie de cambios de rutas genera una solución de mejor calidad, se define esta como la mejor solución actual.

Finalmente se tiene una condición que revisa si se han realizado un 75 % de las iteraciones totales y que no se haya entrado a esta condición previamente, esto gatilla la ejecución del procedimiento *CFiller* definido en la sección 3.6.3. Este procedimiento se encarga de vaciar una ruta que no sea de tipo C, y asignará visitas a nodos de tipo C hasta suplir la demanda, buscando así reducir el deterioro generado en la planta para suplir la demanda de este tipo

de leche.

Finalmente, luego de terminada la cantidad de iteraciones definidas inicialmente, se retorna la mejor solución almacenada como la solución del algoritmo.

Algorithm 1 Iterated Local Search

```
while n < iteraciones
    H <- holgura aleatoria entre un 70% y 100% de la capacidad del camión
    solución <- feaseiblesolution()
    mejor_solución <- solución_inicial
    firstC <- False //controla la ejecución de CFiller
    while m < pasos
        for cada ruta
            solución <- 2opt-neigborhood
            if calidad(solución)>calidad(mejor_solución)
                mejor_solución <- solución
        solución_prev <- solución
        do
            for cada ruta a
                for cada ruta b
                    solución<- Ex-neigborhood(Ruta a, Ruta b)
                    if calidad(solución)>calidad(mejor_solución)
                        mejor_solución <- solución
        while calidad(solución) > calidad(solución_prev)
        max_intentos <- 2 * cantidad de rutas;
        intentos <- 0
        while intentos < max_intentos
            ruta <- ruta aleatoria de la solución actual
            if ruta deteriora la mayoría de su leche
                Purify(solución, solución->routes[s])
                intentos = max_intentos
            intentos <- intentos +=1
        if calidad(solución) > calidad(mejor_solución)
            mejor_solución <- solución
        solución <- AddCandidates(solución)
        if calidad(solución) > calidad(mejor_solución)
            mejor_solución <- solución
        m <- m+1
    if calidad(solución) > calidad(mejor_solución)
        mejor_solución <- solución
```

```
    random <- valor aleatorio entre 0 y 1
    if random < probabilidad de perturbación
        solución <- mejor_solución
        AddCandidates(solucion, 2)
        for i=0 hasta i < factor de perturbación
            rutaA <- ruta cualquiera
            rutaB <- ruta cualquiera distinta a rutaA
            nodoA <- nodo aleatorio de rutaA
            nodoB <- nodo aleatorio de rutaB
            rutaA <- rutaA con el viaje a nodoA intercambiado por nodoB
            rutaB <- rutaB con el viaje a nodoB intercambiado por nodoA
            solución <- solución actualizada con el intercambio de nodos
        if calidad(solución) > calidad(mejor_solución)
            mejor_solución <- solución
    if n >= 0.75*iteraciones y firstC = 0
        for i=0 hasta i< 2 * cantidad de rutas
            solución <- mejor_solución
            solución <- CFiller(solución)
            for cada ruta de solución
                solución <- 2opt-neigborhood
            if calidad(solución) > calidad(mejor_solución)
                firstC <- 1
                i <- 2 * cantidad de rutas
                mejor_solución <- solución
    solución <- reiniciar solución
    n <- n+1
return Solución
```

3.4. Construcción

El algoritmo de construcción en una primera etapa considera una estrategia *greedy* con una inicialización aleatoria respecto al primer nodo visitado por cada ruta. El proceso de construcción de soluciones iniciales prioriza la satisfacción de las demandas de leche de mayor calidad primero, es decir de leche de tipo A, luego de tipo B y finalmente de tipo C. Esto pues la leche recolectada solo puede ver su calidad degradada al realizar mezcla, por lo que para satisfacer la demanda de leche tipo A es necesario recolectar lo suficiente de leche de tipo A y no existe modo de satisfacer dicha demanda mezclando otros tipos de leche. Por su parte, para satisfacer la demande de leche de tipo B es posible utilizar leche de tipo A sobrante al realizar mezcla. De igual manera la demanda de leche tipo C puede se suplida con leche sobrante de tipo A como tipo B.

El algoritmo de construcción de soluciones propuesto busca generar soluciones factibles, pero no asegura que siempre se consigan. Así, se define un parámetro adicional que se denominará *demanda real* para cada tipo de leche. La demanda real se calcula considerando que es posible mezclar posteriormente en planta para suplir la demanda de cada tipo de leche, como se mencionó anteriormente. Para esto, se considera la demanda de leche tipo A como la demanda real de la misma. Esto, dado que esta no puede ser suplida por ningún otro tipo de leche. Posteriormente se revisa si existe la suficiente leche disponible de tipo B para satisfacer la demanda especificada para la misma. De no ser así, la única forma de satisfacer dicha demanda es mezclando leche tipo A con leche tipo B, por lo tanto, se suma lo faltante a la demanda real de tipo A. Posteriormente, para la demanda real de tipo C, al igual que en el caso de leche tipo B, si no es posible suplir la demanda con lo disponible de leche tipo C, esto se suplirá mezclando leche tipo B y de no ser posible, con leche de tipo A.

Posteriormente se comienza a asignar las granjas a visitar a cada camión comenzando por las granjas que producen leche de mejor calidad. Esto hasta que se cumpla alguna de tres condiciones. (1) suplir la demanda real del tipo de leche actual, (2) no queden granjas por visitar del tipo especificado ó (3) o si existe la suficiente leche disponible en los nodos disponibles para visitar para llenar un camión hasta cierto nivel. Esta *holgura* (H) se considera un

parámetro del algoritmo de construcción que será controlado durante el proceso de búsqueda. Una vez cumplida cualquiera de las condiciones, se pasa al siguiente tipo de leche de mejor calidad hasta que no queden camiones disponibles.

Para la selección del siguiente nodo a visitar en cada paso se genera una lista de opciones factibles respecto a capacidad del camión y tipo actual de leche. La idea de esta selección es generar ruta puras respecto al tipo de leche transportado.

Dichos nodos se ordenan en orden ascendente de acuerdo a la distancia al nodo actual, y se escoge su siguiente visita más cercana.

El pseudocódigo del algoritmo 2 muestra el proceso de construcción de soluciones iniciales. Cada solución es inicializada con una cantidad fija de rutas vacías, cuya cantidad corresponde al número de camiones disponibles de la instancia. Posteriormente se calcula la demanda real de cada tipo de leche a recolectar. Luego se comienza a asignar viajes a cada recorrido. Se comienza así por visitar nodos que son de calidad A, luego B, y finalmente tipo C. Para asignar los nodos a visitar, en cada recorrido se genera una lista de opciones que contienen todos los viajes posibles de realizar desde el nodo actual hasta cualquier otro nodo no visitado y se escoge aquel de la lista que está a menor distancia del nodo actual. En caso de corresponder al primer viaje de la ruta la elección del nodo se realiza de manera aleatoria. En el caso de que el camión ya no pueda visitar más nodos productores dado que no posee capacidad para cargar más leche, este retornará a la planta. Además si no quedan camiones disponibles o no existe una cantidad mínima de leche disponible para recolectar de forma de llenar el camión con una cierta holgura (H), se cambia de camión y de tipo de leche. Al terminar de iterar por sobre todos los tipos de leche y por todos los camiones disponibles, se tendrá finalmente un conjunto de rutas para todos los camiones disponibles en la instancia.

Algorithm 2 Construcción de soluciones iniciales

```
function feasibleSolution
    solución <- crear n rutas de camiones vacías
    real <- calcular demanda real por tipo
    for cada tipo de leche requerida
        while demanda real del tipo actual no satisfecha o camión con capacidad
            opciones <- arcos que comienzan por el nodo actual hacia
            nodos no visitados
            if primer viaje de la ruta del camión
                camión[i] <- añadir opción aleatoria
                marcar nodo como visitado
            else
                camión[i] <- añadir opción mas cercana al nodo actual
                marcar nodo como visitado
            if camión[i] esta lleno
                camión[i] <- añadir viaje a planta
                i += 1
            if no hay camiones disponibles o disponible del tipo < (1-H)*capacidad
                    i += 1
                    break
    retornar solución
```

3.5. Búsqueda local

Para mejorar la calidad de las soluciones inicialmente obtenidas, se utiliza un movimiento 2-opt con un enfoque de primera mejora en cada ruta, buscando reducir los costos de transporte dentro de la misma. Además se utiliza un segundo movimiento que realiza intercambio de subrutas también en un enfoque de primera mejora. Ambos métodos serán explicados en profundidad a continuación.

3.5.1. 2-opt

2-opt es una heurística de búsqueda local básica para los problemas de tipo del vendedor viajero [5]. En 2-opt se realiza una operación simple para lograr un mejoramiento en la solución actual de la ruta, la cual es una solución que ya realiza un recorrido establecido.

En este trabajo se utiliza el movimiento base realizado en 2-opt, el cual consiste en eliminar dos arcos de la ruta actual, dividiendo esta en tres partes, y posteriormente re-conectando el segmento intermedio que se separó, pero en orden inverso.

En el ejemplo de la figura 3.3 se ilustra la aplicación del movimiento. Para esto se muestra una ruta inicial, a la cual se le realiza un movimiento 2-opt entre los nodos 2 y 5. Esto implica una ruptura del arco que une los nodos 1 y 2, como también una ruptura del arco que une los nodos 5 y 6. Dado lo anterior, la segunda secuencia de la figura ilustra las tres sub-rutas generadas. La sección de la ruta entre los nodos 2 y 5, la cuál está marcada de color rojo en la figura, es invertida por el movimiento 2-opt. Posteriormente se vuelve a unir la sub-ruta invertida, resultando en la última secuencia de la figura.

$$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$$
$$0 \rightarrow 1 \quad 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \quad 6 \rightarrow 7$$
$$0 \rightarrow 1 \rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 6 \rightarrow 7$$

Figura 3.3: Ejemplo de movimiento 2-opt.

Se utilizó el movimiento 2-opt para generar un vecindario de nuevas rutas, las cuales se comparan respecto a la distancia recorrida. Cada vez que se encuentra una mejor solución se actualiza inmediatamente la solución actual es un esquema primera mejora. Esto se realiza para cada una de las rutas del problema, para buscar así reducir los costos de transporte de manera general de la solución obtenida. Para esto se aplica 2-opt a cada ruta como se muestra en el algoritmo 3, en donde se evalúa la posibilidad de realizar el movimiento en nodos en distintas posiciones, hasta que se encuentre uno que genere una *primera mejora* respecto a la distancia.

3.5.2. Intercambio de subrutas

Para poder mejorar las soluciones iniciales obtenidas con el algoritmo 2, se propone un movimiento que realiza intercambio de subrutas, entre dos rutas de la solución, permitiendo así generar nuevas combinaciones de recorridos. Esto generará cambios en cuanto a la distancia recorrida en la solución, y también puede variar la calidad y cantidad de la leche

Algorithm 3 Vecindario 2-opt

```
function 2opt-neigborhood
    ruta <- ruta previamente establecida
    while indice1 recorre todas las visitas desde un punto aleatorio
        while indice2 recorre todas las visitas desde un punto aleatorio
            if |indice1 - indice2| > 2
                if 2-opt(indice1, indice2) reduce la distancia de la ruta
                    ruta <- 2-opt(indice1, indice2)
            indice2 += 1
        indice1 +=1
    retornar ruta
```

recolectada en cada ruta. Esto debido a la posibilidad de realizar un intercambio se produzca una mezcla de distintas calidades en el camión, teniendo posibles deterioros en la calidad de alguna de estas.

Para este movimiento de intercambio se consideran dos rutas iniciales a las que se les realizará un cambio. Para cada una de ellas se fija un índice inicial (i_a y i_b)y un tamaño que determina la cantidad de nodos a intercambiar entre ambas rutas (l_a y l_b, respectivamente). Por ejemplo, en la figura 3.4 se realiza un movimiento de intercambio que comienza en el índice 1 de ambas rutas (k_1 y k_2), y considera $l_a = 2$ y un $l_b = 3$. Esto quiere decir que, en la primera ruta se dejan de visitar dos nodos, pero dichas visitas se cambiaran por tres visitas correspondientes a la segundad ruta. Así, se remueven las visitas 2 y 3 de la primera ruta y se agregan las visitas a los nodos 6, 7 y 8 en ese punto. Por otro lado, a la segunda ruta se le remueven las vistas a 6, 7 y 8 y se le agregan las visitas a 2 y 3 que fueron removidas de la primera ruta. La figura 3.4 muestra las dos rutas resultantes del proceso de intercambio explicado k'_1 y k'_2.

Este movimiento solo realiza el movimiento de intercambio entre rutas si dicho movimiento es factible en cuanto a capacidad de los camiones para ambas rutas. A partir de este movimiento es posible generar un vecindario de nuevas soluciones. La función utilizada para generar el vecindario 4 evalúa todos los puntos de inicio considerando intercambios de 1 a 3 de nodos a intercambiar. El movimiento acepta inmediatamente los cambios que mejoran las soluciones en un esquema primera mejora.

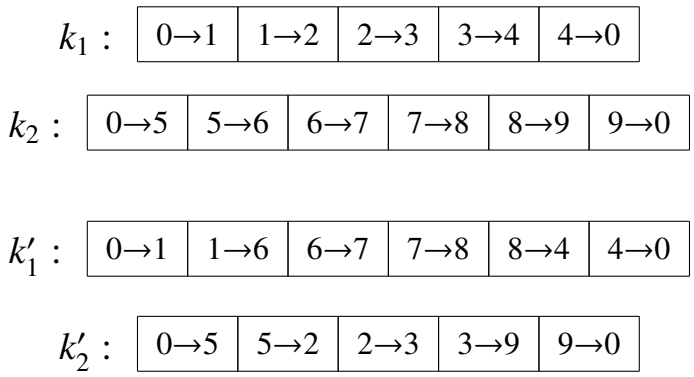


Figura 3.4: Ejemplo de movimiento de intercambio.

Algorithm 4 Vecindario Intercambio

```
function Ex-neigborhood
    RutaA <- Una ruta de la solución
    RutaB <- Una ruta distinta de la solución
    for subrutas A de largo 1 a La
        for subrutas B de largo 1 a Lb
        while indice1 recorre todas la visitas de un punto aleatorio de la rutaA
            while indice2 recorre todas la visitas de un punto aleatorio de la rutaB
                if hacer cambio de subrutas mejora la solución
                    Exchange(RutaA,ia,La,RutaB,ib,Lb)
                    retornar solución actualizada
                indice2 += 1
            indice1 += 1
```

3.6. Perturbaciones

Con la finalidad de poder diversificar en el espacio de búsqueda de soluciones se implementaron algoritmos de perturbación que modifican la solución en la que se esté trabajando.

Para realizar lo anterior se utiliza una solución a la que se le hayan aplicado los movimientos de intercambio de subrutas y 2-opt, se aplica un movimiento llamado purificar, el cuál se encarga de tomar una ruta aleatoria y remover las visitas a granjas de la peor calidad presente en la ruta.

Luego de realizar esta primera perturbación se utiliza el movimiento de agregar candidatos para rellenar las rutas que han pasado por el proceso de purificación. Finalmente, se utiliza CFiller, el cuál se se encarga de vaciar una ruta aleatoria de la solución, y solo visitar nodos de tipo C, para intentar satisfacer la demanda de esta sin reducir la calidad de los otros tipos de leche recolectada.

3.6.1. Purificación de mezclas

Este procedimiento busca mejorar la calidad de una ruta de la solución actual removiendo visitas a granjas culpables del deterioro de la leche del camión.

El movimiento selecciona una ruta de la solución actual, y revisa la calidad actual de esta. Posteriormente se calcula el porcentaje de leche de la calidad actual de la ruta respecto al total cargado en el camión, es decir, si una ruta es de calidad B y transporta 100L de leche, y de estos solo 10L fueron adquiridos de un nodo de calidad B, corresponde a un 10 % de leche B del total. Posteriormente se verifica que este monto sea igual o menor al 10 % del total recolectado por el camión y se procede a remover todas las visitas a predios de esa calidad, mejorando así la calidad de la ruta.

Este movimiento será ejecutado una cantidad definida de veces, proporcional a la cantidad de rutas de la solución, y será ejecutado siempre, luego de una vez obtenido la solución optimizada por 2-opt.

Algorithm 5 Purificación de mezclas

```
function Purify
    Ruta <- Una ruta de la solución
    for Calidad in [A,B,C]
        if Litros(Calidad)/Total <= 0.1
            for viaje in Ruta
                if nodo destino de calidad == Calidad
                    remover viaje
    return ruta actualizada
```

3.6.2. Agregar candidatos

Este procedimiento busca generar una perturbación en la solución actual, de tal manera que remueva visitas a granjas desde la solución actual y añada, de igual forma, nodos que no fueron visitados inicialmente a las distintas rutas.

El movimiento consta de dos etapas. En la primera etapa se seleccionan al azar, de una lista de nodos no visitados por la solución, todos los nodos disponibles para visitar como candidatos a incorporar a las rutas. Posteriormente, en la segunda etapa, se trata de agregar cada uno de los nodos candidatos seleccionados a las rutas actuales, considerando que estos sean del mismo tipo de leche, o en el caso de una ruta vacía se inicializa con el candidato actual como se muestra en la figura 6.

Algorithm 6 Agregar candidatos

```
function AddCandidates
    candidatos <- Se seleccionan nodos no visitados al azar
    Ruta <- Una ruta de la solución
    for candidatos
        indice<- suma entre i y un desplazamiento aleatorio
        i = 0
        while i < tamaño de la ruta
            if la ruta tiene capacidad para añadir al candidato y es del mismo tipo
                Ruta<- ruta con el candidato ingresado como primer viaje
                break
            else if la ruta es de tamaño 0
                Ruta<- ruta con el candidato ingresado como primer viaje
            i++
    return rutas actualizadas
```

Este movimiento de perturbación siempre se ejecuta posterior al movimiento de purificación de la sección anterior, y también puede ser ejecutado posteriormente dada la probabilidad de perturbación definida en las variables, en ambas circustancias esto modificaría la solución actual.

3.6.3. Incremento de recolección de leche C

Este procedimiento busca reducir el deterioro de leche de mejor calidad que se produce al recolectar leche de alta calidad, para luego transformarla en leche tipo C en la planta.

Este movimiento realiza tres procesos como se muestra en el pseudocódigo del algoritmo 7. En primer lugar genera una lista de nodos candidatos de calidad C que no hayan sido visitados, y se almacenan en una lista ordenada de manera aleatoria. Posteriormente se elige una ruta al azar de la solución y se vacía. Probablemente conteniendo solo leche de tipo A y B. Finalmente se añaden visitas desde la lista ordenada de manera aleatoria a esta ruta hasta que se complete la capacidad del camión.

Algorithm 7 Incremento de recolección de leche C

```
function CFiller
    candidatos <- Se seleccionan nodos de calidad C no visitados al azar
    Ruta <- Una ruta al azar de la solución
    Ruta = Ruta vacía
    for Candidato en candidatos
        if candidato puede ser añadido a la Ruta
            Ruta <- añadir candidato
        else
            Ruta <- visita final a la planta
    return rutas actualizadas
```

3.7. Resumen

El algoritmo implementado para este trabajo consta de tres etapas: Construcción de una solución inicial, optimización de soluciones y perturbaciones de estas.

El proceso de construcción de una solución inicial considera la generación de rutas priorizando recorrer predios de mayor calidad y generando soluciones factibles en su mayoría, no cumpliendo en todos los casos con la *demanda real* de cada tipo de Leche. Posteriormente se utilizaron dos movimientos cuyo objetivo es modificar las rutas actuales. Primero 2-opt reordena el orden de visita de los nodos de una ruta, para optimizar así las distancias recorridas. En segundo lugar se intercambian subrutas de distintos pares de estas para generar nuevas combinaciones de recorrido. Finalmente se realizan perturbaciones para expandir más aún el espacio de búsqueda de nuevas rutas posible, tratando así de poder acercarse más a un máximo global,

Capítulo 4

Experimentos

4.1. Instancias

Con el objetivo de evaluar las técnicas propuestas, se consideraron dos conjuntos de instancias de pruebas. El primer conjunto, que denominaremos instancias pequeñas o de pruebas, corresponden a un subconjunto de 40 nodos pertenecientes a la instancia utilizada por Paredes-Belmar [16] la cual posee un total de 500 nodos. El segundo conjunto de instancias, denominado instancias reales, considera un grupo de instancias obtenidas a partir de la información de la distribución de los 500 nodos, pero de mayor tamaño generadas a partir de un proceso de agrupamiento de nodos de modo de conseguir instancias de mayor tamaño. En este conjunto también se consideran instancias de prueba que trabajan con los 500 nodos.

Para ambos conjuntos el costo de transporte por kilómetro recorrido C equivale a $1u.m.$. Además los tres tipos de leche presente en el problema tienen valores de venta $L^A = 0.03u.$, $L^B = 0.021u.$, $L^C = 0.009u.$.

4.1.1. Instancias de prueba

En la tabla 4.2 se pueden ver numeradas cada una de las distintas instancias, en donde la primera columna presenta la cantidad de camiones disponibles para cada instancia, la segunda tiene la capacidad homogénea de los camiones en litros. Además cada columna Q_x corresponde a los litros demandados de cada calidad de leche. Por último las tres últimas columnas representan el porcentaje de leche demandada respecto a la disponible para cada una de las calidades.Por ejemplo, para la primera instancia se demanda un 98 % de la leche de calidad A disponible.

Instancia	Camiones	Capacidad	p_a	p_b	p_c	$p^a/Disp_a$	$p^b/Disp_b$	$p^c/Disp_c$
t1	7	20000	60000	40000	10000	98 %	70 %	13 %
t2	5	30000	60000	40000	10000	98 %	70 %	13 %
t3	4	30000	0	0	0	0 %	0 %	0 %
t4	5	30000	0	0	0	0 %	0 %	0 %
t5	5	30000	10000	70000	10000	16 %	123 %	13 %
t6	5	20000	40000	40000	10000	65 %	70 %	13 %

Tabla 4.1: Detalle instancias de prueba.

Cada una de estas seis instancias corresponden a un conjunto de 40 nodos, los cuales tienen las mismas ubicaciones geográficas, cantidades de producción y calidad de producción de cada una. Lo que genera diferencia entre estas son las restricciones para el problema de optimización, como la cantidad de camiones disponibles y su capacidad máxima de recolección, y en la cantidad mínima demandada por la planta para cada tipo de leche.

Se observa que de las seis instancias, existen dos las cuales no tienen una demanda mínima de leche a recolectar, por lo que tienen libertad en cuanto de cada una se debe recolectar, limitado por las restricciones de capacidad de los camiones de todas formas. Además también se aprecia que existe una instancia que requiere recolectar el 123 % de la leche de tipo B disponible, por lo que muestra que será necesaria hacer mezcla para lograr suplir la demanda.

4.1.2. Instancias reales

Para las instancias reales, se utilizaron 500 nodos de granjas de producción ubicadas en el sur de Chile, utilizando estos se generaron seis conjuntos, siendo el primero el original y los otros cinco se obtuvieron mediante la utilización de un algoritmo *k-means* [13].

El objetivo del algoritmo previamente mencionado es el de agrupar datos similares, a través de la comparación de características comunes o patrones de estas. Para esto *k-means* busca generar una cantidad k de agrupaciones de datos, generando en cada uno un especie de dato central y se consideran dentro de esa agrupación los datos que estén una distancia cercana a este centro.

Instancia	Nodos	Camiones	Capacidad	p_a	p_b	p_c	$p^a/Disp_a$	$p^b/Disp_b$	$p^c/Disp_c$
r1	128	9	30000	35000	195000	15000	17 %	123 %	12 %
r2	143	9	30000	20000	200000	15000	15 %	124 %	13 %
r3	142	9	30000	25000	190000	20000	18 %	121 %	15 %
r4	31	3	30000	8997	41658	8634	18 %	123 %	13 %
r5	56	6	30000	15000	140000	10000	19 %	122 %	13 %
r500	500	35	30000	80000	800000	750000	15 %	119 %	128 %

Tabla 4.2: Detalle instancias reales.

En la tabla 4.2 se describen las características de las instancias de pruebas construidas a partir del total de nodos, utilizando *k-means*.Para cada una se muestra la cantidad de nodos de cada instancia, dado que a diferencia de las instancias de pruebas esto varía. Además de la cantidad de camiones disponibles, la capacidad máxima de estos en litros, y la cantidad de litros requeridos de cada calidad de leche, también incluyendo las últimas tres columnas previamente explicadas que definen el porcentaje de leche de cada tipo demandada respecto al total disponible para cada instancia.

Se observa que de las seis instancias, que la cantidad de nodos en estas varían de 31 hasta los 500 que representan el total, y que la cantidad de camiones disponibles varía de 3 a 35. Además que la cantidad de leche requerida para cada tipo también varía para cada uno de estos subgrupos del conjunto total.

4.2. Sintonización de parámetros

El proceso de sintonización tiene como objetivo determinar los valores óptimos para los parámetros del algoritmo antes de su ejecución. En este trabajo de memoria, se utilizó el método de sintonización ParamILS, basado en la sintonización manual, como se propuso en [18]. ParamILS es un enfoque automático que emplea una búsqueda local iterativa para encontrar la configuración óptima de parámetros para un algoritmo metaheurístico, para esto se emplea un conjunto de prueba como conjunto de entrenamiento.

Para entrenar ParamILS, se realizaron un máximo de 25000 ejecuciones del algoritmo utilizando como conjunto de entrenamiento las instancias de pruebas definidas previamente en la sección 4.1.1. Además, se definió el conjunto de valores posibles y un valor inicial para el proceso de sintonización, los cuales están registrados en la Tabla 4.3.

Parámetro	Valores Posibles	Valor Inicial
Probabilidad de Perturbación	0.01, 0.05, 0.1, 0.15, 0.2	0.05
Factor de Perturbación	1, 2, 3, 5	2
P	1000, 10000	1000
Pasos	50, 100, 200, 500, 1000	200

Tabla 4.3: Resultado sintonización de parámetros.

El primer parámetro, como su nombre indica, se utiliza para definir la probabilidad de ejecutar el movimiento de perturbación llamado *AddCandidates*, tal como se describe en la sección 3.6.2. Este movimiento puede ejecutarse al final de cada iteración con una probabilidad igual al valor del parámetro. Además, dentro de esta sección, se realizarán intercambios aleatorios de nodos entre rutas. Es importante destacar que la ejecución de estos dos movimientos siempre modificarán la mejor solución.

Es crucial tener en cuenta que el primer parámetro mencionado anteriormente define cuándo se activa la fase de perturbación del algoritmo. Esta sección solo se ejecutará si se cumple la condición especificada por la probabilidad definida en el primer parámetro, la cual puede variar entre un mínimo del 1 % y un máximo del 20 %. Durante la sintonización de parámetros, se determinó que el valor óptimo para este parámetro es del 5 %.

El siguiente parámetro define el factor de perturbación, que controla la cantidad de nodos

que se intercambiarán al entrar en la sección activada por la probabilidad del primer parámetro. Los nodos y las rutas que los contienen se seleccionan de forma aleatoria para realizar estos intercambios. La cantidad de intercambios puede variar entre un mínimo de una vez y un máximo de cinco. Sin embargo, después de la sintonización de parámetros, se encontró que la configuración óptima implica realizar dos intercambios.

El parámetro referido como p es el definido en 3.2, define el valor que toma la variable P del término F^{tr} definido en la ecuación 3.2, y utilizado en la función objetivo 3.1. Este define la magnitud en la que el déficit de leche recolectada castiga a la calidad de la solución. Esta variable, al buscar magnificar más que nada el orden de magnitud de la solución, solo podía tomar los valores 1000 y 10000, y siendo el óptimo para evaluar el primero.

Finalmente la variable denominada 'Pasos' en la tabla 4.3 define cuántas veces se ejecuta el conjunto de movimientos de mejora de la solución, ejecutándose en una primera instancia en la solución inicial, y luego en alguna solución mejor a estar obtenida por alguna iteración. El proceso completo se ejecuta un máximo de veces definido por el parámetro. El valor óptimo de intentos de mejora obtenida por el sintonizador de parámetros fue de 200, evaluando desde un mínimo de 50 intentos hasta un máximo de 1000.

4.3. Entorno experimental

Cada una de las pruebas de este trabajo de memoria fue ejecutada con 20 semillas diferentes, tratando así de amortizar el efecto de la naturaleza estocástica del algoritmo propuesto. Las pruebas presentadas fueron realizadas en un servidor Power Edge R630 con 2 CPUs Intel(R) Xeon(R) E5-2680 v3 @ 2.50GHz, 64 GB de RAM corriendo bajo distribución Ubuntu x64 16.10.

4.4. Resultados

En esta sección se presentan los resultados que se obtuvieron durante el proceso de evaluación del algoritmo propuesto en esta memoria. Primero se presentarán los resultados obtenidos para las instancias de prueba, luego los resultados para las instancias reales. Además se compararán estos con los obtenidos por un *solver* que utiliza un acercamiento completo, y también con soluciones propuestas previamente para estas mismas instancias.

Para todas estos métodos de resolución los resultados presentados se obtuvieron utilizando la función de evaluación presentada en la ecuación 1.1.3.

4.4.1. Comparación en instancias de prueba

En este inciso se presentan los resultados comparando las soluciones obtenidas del algoritmo propuesto, con los obtenidos por el *solver* CPLEX, también los resultados obtenidos para las mismas instancias por el algoritmo basado en *Simulated Annealing* propuesto por [19], el cuál será referido como **MCSA** y también el algoritmo de dos fases propuesto en [15], abreviado como **2PRA**.

Instancia	CPLEX	MCSA	2PRA		ILS				
	Best	Best	Best	Avg	Best	Avg	Gap CPLEX	Gap MCSA	Gap 2PRA
t1	750.3	**716.9**	675.0	484.0	570.4	466.0	-31.5	-25.7	-18.3
t2	969.4	**948.4**	877.0	746.0	918.7	890.6	-5.5	-3.2	4.5
t3	1254.3	**1249.3**	1239.0	1231.0	1176.9	1112.1	-6.6	-6.1	-5.3
t4	1311.0	1230.0	**1296.0**	1295.0	1210.42	1154.5	-8.3	-1.6	-7.1
t5	1240.6	1003.0	**1241.0**	1234.0	1241.0	927.2	-17.9	4.7	-18.0
t6	604.1	**604.4**	604.0	595.0	330.7	7.4	-82.7	-82.7	-82.6

Tabla 4.4: Detalle resultados instancias de prueba.

Para todos los resultados tabulados en la tabla 4.4 la solución óptima siempre se obtiene por el *solver*, ya que este utiliza un enfoque completo para resolver las instancias, así que en la tabla se resalta, en negrita, la solución óptima para cada instancia, obtenida por los algoritmos de búsqueda incompleta.

Para la instancia t1 se tiene GAPs negativos con la solución obtenida por CPLEX de un 31.5 %, con MCSA posee un 25.7 % de diferencia, y finalmente presenta una solución un

18.3 % de peor calidad que la propuesta por 2PRA.

Similarmente, en la instancia t2, se evidencian GAPs negativos de un 5.5 % y de 3.2 % con los resultados obtenidos por CPLEX y MCSA respectivamente, sin embargo, el algoritmo presenta una solución un 4.5 % mejor que la presentada por 2PRA.

En t3 el algoritmo propuesto, vuelve a obtener una solución que es inferior a las propuestas en trabajos previos, siendo un 6.6 % peor al óptimo, un 6.1 % a la que entrega MCSA y un 5.3 % con 2PRA.

Para t4, ILS vuelve a presentar solo soluciones inferiores, siendo de una calidad 8.3 % inferior a la obtenida por CPLEX, un 1.6 % comparado con MCSA y 7.1 % menor a 2PRA.

Para la t5, presenta una solución menos óptima a CPLEX y 2PRA GAP de 17.9 % y 18 % respectivamente, aunque presenta una mejora de un 4.7 % en comparación al resultado de MCSA.

Para la instancia t6, la solución propuesta en esta memoria presenta la mayor diferencia de entre todas las instancias utilizadas, siendo un 85.7 % menos óptima en comparación a todos los otros métodos comparados.

El mejor resultado obtenido por el algoritmo es para la instancia t2, en el cual presenta un resultado más cercano al *solver* y también supera al resultado obtenido por 2PRA.

En el caso contrario, el peor resultado obtenido corresponde al obtenido para la instancia t6, donde este presenta diferencias significativas con todos los otros métodos. Aunque es importante notar que esta es la instancias de prueba más desafiante a resolver, ya que la configuración presenta solo un 10 % de holgura respecto de lo demandado y la capacidad total de los camiones.

En cuanto a tiempos de ejecución la tabla 4.5 presenta los tiempos promedio para cada instancia y cada algoritmos respectivamente, contrastando así también con el algoritmos CPLEX que fue ejecutado con un tiempo límite de 30 minutos. Dado lo anterior, los trabajos previos y el acercamiento propuesto en esta memoria en promedio son menores, esto pues no son métodos de resolución completa. Así, MCSA dependiendo de la instancia, puede ser

Instancia	CPLEX	MCSA	2PRA		ILS	
	Max	Avg	Best	Avg	Best	Avg
t1	1800.0	235.0	171.0	193.0	7.7	**6.8**
t2	1800.0	142.0	143.0	223.0	6.1	**6.2**
t3	1800.0	**32.0**	103.0	278.0	101.1	108.1
t4	1800.0	**88.0**	110.0	285.0	154.4	141.1
t5	1800.0	**242.0**	224.0	303.0	100.6	703.6
t6	1800.0	141.0	17.0	188.0	2.0	**2.0**

Tabla 4.5: Tiempos de ejecución para instancias de prueba.

desde un 86.5 % a un 98.2 % más rápido, en el caso de 2PRA desde un 83.1 % hasta un 89.5 % en promedio, y finalmente ILS que logra tardar desde un 60.9 % hasta un 99.8 % menos tiempo en ejecutarse en promedio.

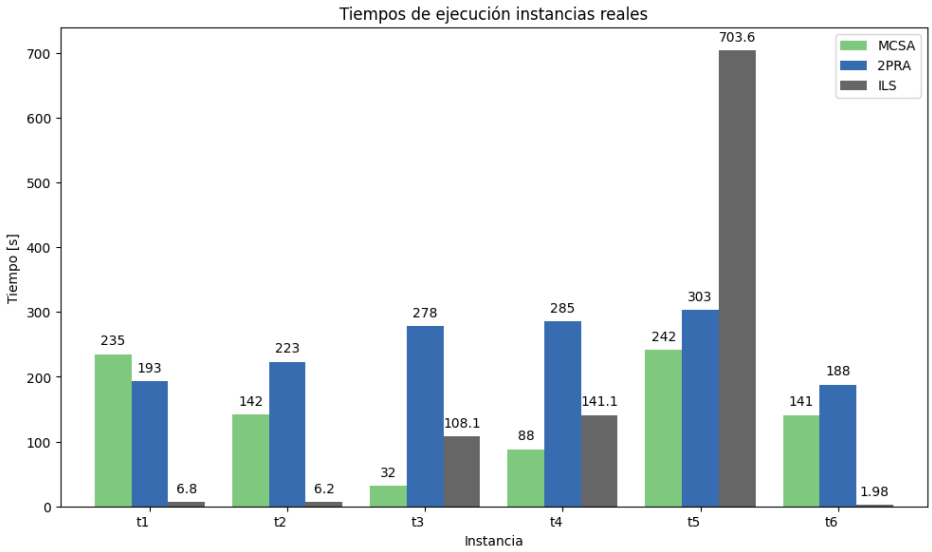


Figura 4.1: Tiempos Promedios para instancias de prueba.

Como se puede apreciar en el gráfico de la figura 4.1 los tiempos de los tres algoritmos suelen variar por instancia. Esto, dado que tienen condiciones de parada, o una limitación en la cantidad de iteraciones en contraste al *solver* que define un tiempo límite máximo.

En la figura 4.1 se puede ver que algoritmo ILS propuesto, suele presentar tiempos de ejecución inferiores en promedio, como se da el caso para las instancias t1, t2 y t6. Sin embargo, para las instancias t3 y t4 presenta mejores tiempos que el algoritmo 2PRA pero toma más tiempo que MCSA. Finalmente, para la instancia t5 presenta el peor rendimiento

en cuanto a tiempo, ya que en promedio demora más del doble que los otros algoritmos comparados.

Si bien el algoritmo propuesto suele tener tiempos de ejecución menores o en el orden de magnitud de los otros que se han usado para comparar, este tiempo no tiene relación en sí con los resultados, por ejemplo, en las primeras dos instancias los tiempos son inferiores a siete segundos, el cuál es mucho menor a los otros dos algoritmos, pero la calidad de la solución es mucho menor para la primera, entre 25.7 % y 18.3 % de peor calidad, aunque en la segunda, este resulta entregar un resultado mejor al obtenido por 2PRA, pero no así para MCSA.

Como un análisis extra se puede decir que el algoritmo propuesto no es bueno en general para instancias pequeñas, especialmente cuando se tiene poca holgura para la demanda respecto al total, y con una cantidad muy limitada de camiones, esto debido a que la construcción de soluciones inicial puede estar limitando mucho el espacio de posibles soluciones a recorrer.

4.4.2. Resultados de instancias reales

Para las instancias reales se comparan las soluciones obtenidas del algoritmo propuesto, con las generadas por MCSA y 2PRA. En este caso no se comparará con CPLEX dado que, al ser instancias grandes, este solver no es capaz de obtener soluciones en tiempos acotados.

En la tabla 4.6 se presentan los resultados obtenidos para los tres algoritmos, teniendo así para la primera instancia, que al igual que el resto son un subconjunto de la instancia r500, como se explica en la sección 4.1.2.

Instancia	MCSA	2PRA		ILS			
	Best	Best	Avg	Best	Avg	Gap MCSA	Gap 2PRA
r1	5018.0	4966.0	4931.0	**5742.1**	5658.0	12.6	13.5
r2	3884.4	3881.0	3846.0	**4206.7**	4159.2	7.7	7.7
r3	3496.6	3578.0	3516.0	**3997.5**	3938.9	12.5	10.5
r4	1307.0	1251.0	1251.0	**1421.4**	1398.6	8.1	12.0
r5	3017.7	**3019.0**	2994.0	2927.1	2836.3	-3.1	-3.1
r500	16531.2	17637.0	17504.0	**20807.4**	20663.9	20.6	15.2

Tabla 4.6: Detalle resultados instancias reales.

Para la instancia r1, el algoritmo propuesto genera en promedio mejores resultados a los obtenidos por los otros dos algoritmos, y obtiene en la mejor ejecución una configuración que es 12.6 % superior a MCSA y un 13.5 % respecto de 2PRA.

Para la instancia r2, nuevamente el algoritmo ILS propuesto logra obtener el mejor resultado, obteniendo valores que superan en un 7.66 % y 7.74 % a MCSA y al denominado 2PRA.

En la instancia r3 también logra presentar mejoras de un 12.5 % respecto de MCSA, también un incremento de la calidad de la configuración de ruta obtenida de 10.5 % comparado a 2PRA para este subconjunto de nodos.

En la instancia r4 presenta mejoras nuevamente respecto a los dos trabajos anteriores, siendo así un 8.047 % mejor que MCSA y un 11.98 % que 2PRA.

Para el caso r5, se rompe esta tendencia ya que los resultados obtenidos por MCSA y 2PRA, son un 3.1 % mejores que los generados por el algoritmo propuesto.

Finalmente para la instancia completa (r500), el método propuesto logra obtener una solución que tiene una calidad un 20.6 % mejor que la obtenida por MCSA, también esta logra ser un 15.2 % mejor respecto a 2PRA.

En la tabla (4.7), se presentan los tiempos de ejecución para obtener el mejor resultado (*Best*) entre todas las ejecuciones, y también el promedio (*Avg*) de estas.

Instancia	MCSA		2PRA		ILS	
	Best	Avg	Best	Avg	Best	Avg
r1	1127.0	1547.0	234.0	**550.0**	1276.2	1244.4
r2	3396.0	1850.0	104.0	**645.0**	1832.7	1813.2
r3	2178.0	2282.5	596.0	**713.0**	1808.7	1811.7
r4	152.0	208.5	75.0	288.0	19.0	**22.0**
r5	549.0	568.0	189.0	347.0	43.2	**35.3**
r500	22769.0	34178.0	465.0	**489.0**	1951.2	1991.1

Tabla 4.7: Tiempos de ejecución para instancias reales.

En cuanto a los tiempos de ejecución, el algoritmo ILS propuesto, logró tiempos promedios mejores que MCSA y 2PRA para las instancias r4 y r5, llegando a ser ,en promedio, hasta un 89.4 % más rápido que MCSA, y un 92.4 % para la instancia r4, para la quinta también logra demostrar una reducción del tiempo de ejecución promedio de 93.8 % comparado a MCSA y un 89.8 % respecto de 2PRA.

Al considerar las otras instancias, tanto las primeras tres instancias como el conjunto completo de nodos, ILS no logra ser más rápido que 2PRA, siendo este último un 44.2 % más rápido que ILS para la instancia r1, además de un 64.4 % y un 60.6 % más rápido que el algoritmo propuesta para la instancias r2 y r3. Sin embargo, la propuesta logra ejecutarse en un menor tiempo que MCSA para estas configuraciones, logrando así tiempos un 19.6 % mejores en promedio para r1, un 2 % en r2, y 20.6 % para r3.

Finalmente para la instancia completa de 500 granjas, 2PRA resulta ser un 75.4 % más rápido que ILS, pero este úlitmo, a su vez, tarda en promedio un 91.3 % menos que MCSA.

Además de los datos tabulados se representaron el gráfico 4.2, donde es importante destacar que se removió del conjunto el tiempo de ejecución promedio de MCSA para la instancia real completa dado que al tener un valor muy superior al resto de los datos, dificulta la comparación con el resto de tiempos.

En las pruebas realizadas con las instancias reales, se observa que en general ILS, suele presentar mejores soluciones para estos conjuntos grandes de nodos a visitar, obteniendo mejores resultados comparado con trabajos previos para todas las instancias a excepción de r5, en la cual presenta un calidad solo un 3.1 % menor a la obtenida por los otros algoritmos.

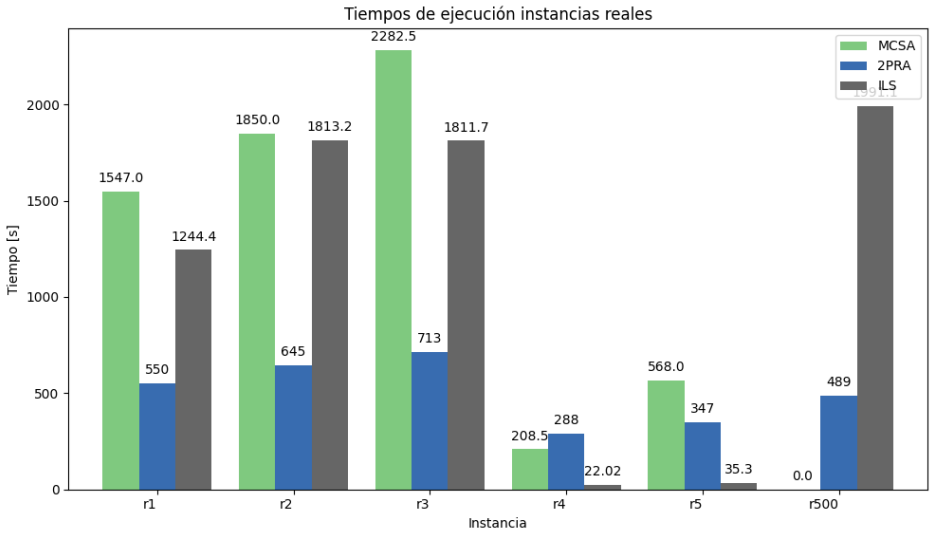


Figura 4.2: Tiempos Promedios para instancias reales.

Esto último puede deberse a la gran cantidad de combinaciones posibles a generar, al ser un número mayor, es más probable que los movimientos de perturbación puedan encontrar una solución que ayude a escapar de un óptimo local y así encontrar una configuración mejor.

4.5. Resumen

En este capítulo se presentaron las instancias que se utilizaron para evaluar las soluciones propuestas en trabajos previos, y el algoritmo ILS implementado. Estas instancias se dividen en dos grupo. Las instancias de pruebas, que son subconjuntos de una instancia real más grande, cuyas configuraciones en cuanto a cantidad de camiones y capacidad se definieron para probar distintos escenarios. Además se presentaron las instancias reales, y algunos subconjuntos del problema cuya configuración es de mayor escala que la de las instancias de prueba.

También se presentó el proceso de sintonización de parámetros que fueron ajustados con las instancias de pruebas, utilizando el método de sintonización de parámetros ParamILS.

Posteriormente se presentaron los resultados para las instancias de prueba, presentando la calidad de las soluciones obtenidas y los tiempos de ejecución del algoritmo ILS propuesto

en estas. Así también se compararon los resultados con los algoritmos MCSA y 2PRA propuestos en trabajos previos. ILS obtuvo en general resultados inferiores en cuanto a calidad siendo hasta un 82.7 % peor y solo un 4.7 % respecto a las soluciones obtenidas por los otros algoritmos.

Finalmente se presentan y comparan las soluciones que obtuvo la propuesta en las instancias reales, comparándolas nuevamente con los obtenidos en trabajos anteriores. El algoritmo propuesto generó soluciones que llegaron a tener una calidad hasta un 20.6 % mejor y en el peor de los casos ha sido solo un 3.1 % inferior.

Conclusiones

Esta memoria presenta una variación de el VRP clásico, la cual se denominó MCPSB, cuyas siglas en ingles refieren a Milk Collection Problem With Selection and Blending. Esta variante del problema considera la existencia de leche de distintos tipos o de distintas calidades, lo cual permite la mezcla de estas tanto en el camión que la recolecta como en la planta de procesamiento donde es recibida finalmente. Además se considera un conjunto de predios productores de leche que suplen un solo tipo de leche, los cuales hay que visitar con la finalidad de cumplir una cuota mínima de cada una de estas, siempre y cuando la selección de nodos y la ruta genere un beneficio respecto a los costos.

Se revisa el estado del arte del problema VRP, y sus variantes que involucran recolección de distintos tipos de elementos. Además se investigaron variaciones relacionadas directamente a la recolección de leche que han sido presentado en otros trabajos de distintos y con condiciones diferentes al presentado. Finalmente todos tienen la misma meta común, lo que implica encontrar las rutas para una flota de vehículos, las cuales visiten un conjunto de nodos en un orden que minimice los costos de transporte y maximice la ganancia que implica la recolección.

Se presenta un método para resolver el MCPSB, utilizando una meta-heurística basada en búsqueda local iterativa que permite buscar en espacios de soluciones infactibles. El algoritmo presenta una fase de construcción inicial que genera una solución que prioriza suplir la demanda de tipos de mayor calidad, en caso de no haber demanda, sigue priorizando la obtención de leche de la calidad más alta.

Posteriormente se presenta una fase de búsqueda local, donde se busca refinar la solución

inicial obtenida en la fase de construcción, mediante un movimiento 2-opt, y también intercambios de subrutas de camiones distintos, buscando así mejorar la solución generada en primera instancia.

En la fase de perturbación se busca diversificar las soluciones de la búsqueda local, para esto se implementó un movimiento purificador, el cuál toma una ruta y si este visita nodos que proveen de leche de más de un tipo, remueve las visitas a aquellos que son de menor calidad. Finalmente se agregó un método *Filler* que busca evitar el deterioro de la leche recolectada al ser mezclada en la planta para cumplir la demanda de la que sea de calidad inferior, para esto se vacía una ruta aleatoria y se utiliza una lista de granjas de peor calidad que no han sido visitadas.

Posteriormente, utilizando el método de sintonización de parámetro ParamILS, se buscaron optimizar los valores los parámetros de probabilidad de aplicar una perturbación a la solución, el factor de perturbación que determina cuántas veces se aplica el método Agregar Candidatos, el factor que se encarga de amplificar el castigo que se le aplica a la función de evaluación.

Luego se evaluó el algoritmo ILS propuesto en dos conjuntos de instancias. Uno sintético y un conjunto de instancias reales.

En cuanto a los resultados, para las instancias de prueba sintéticas, el algoritmo probó tener un desempeño negativo, obteniendo soluciones hasta un 82.7 % de peor calidad que el solver utilizado, y los otros dos trabajos de memoria comparados. En el mejor caso logra una solución de una calidad 5.5 % menor a la obtenida por el método de resolución completa. En contraste, para las instancias reales, el algoritmo propuesto logró obtener mejores resultados que los métodos MCSA y 2PRA comparados, logrando en el peor caso una solución un 3.1 % peor para un subconjunto de nodos de la instancia completa, y para esta llegó a lograr una mejora de hasta un 20.6 % respecto a MCSA y de calidad un 15.2 % superior a 2PRA, además que en general para este último logró obtener esta mejor solución en un tiempo de 32 minutos y 31 segundos, lo cuál es muy inferior a los 6 horas y 19 minutos que le tomó a MCSA pero no inferior a los 8 minutos que toma 2PRA.

Se aprecia que los tiempos de ejecución del algoritmo propuesto para las primeras 3 instancias son mayores que los de 2PRA, pero logra ser ligeramente más rápido que MCSA. Por el contrario, para el cuarto y quinto subconjunto los tiempos promedios de ejecución de la solución propuestas son inferiores a los otros dos métodos.

Para el conjunto completo de nodos, el algoritmo ILS propuesto, logra obtener tiempos promedios mucho menores que MCSA, dado que estos logran ser un 91.4 % menores. En cambio, si se compara a los tiempos promedios de 2PRA, las soluciones en promedio suelen tardar un 75.5 % más en generarse.

Los resultados obtenidos por ILS logran una mejora en comparación a los trabajos previos en la instancia completa, y sus subconjuntos, esto puede tener relación a la cantidad de nodos disponibles y de camiones, también así por la proporción de demanda de cada tipo de lecho respecto al total disponible por tipo para cada configuración definida por instancia.

Finalmente, como un trabajo futuro se podría presentar una mejor manera de generar soluciones iniciales, ya que al priorizar las demandas de tipos de leche de calidad superior primero, se limita mucho el espacio de solución de los que poseen una calidad inferior a la máxima, se puede considerar algún método que considere como prioridad aquel tipo que presenta una demanda porcentual mayor del total. Además como una segunda mejora, el principal elemento que reduce la calidad de las soluciones después de los costos de transporte es el deterioro, dado que al momento de realizar una mezcla se devalúa la solución, lo que en realidad haría que los costos de ir hacia un nodo, el cual redujo la calidad de cierta parte de lo recolectado, involucren un menor beneficio para ese viaje, para esto se podría añadir algún parámetro calculado a cada viaje el cuál considere el transformar la leche de la granja que se visitará, a una de tipo inferior.

Bibliografía

[1] Mark Abkowitz and Paul Der-Ming Cheng. Developing a risk/cost framework for routing truck movements of hazardous materials. *Accident Analysis Prevention*, 20(1):39–51, 1988.

[2] Art 204. Reglamento sanitario de los alimentos. diario oficial de la república de chile. http://www.dinta.cl/wp-content/uploads/2021/03/RSA-DECRETO_977_96_act-02-02-2021.pdf. Accessed: 2021-06-20.

[3] Egon Balas. The prize collecting traveling salesman problem. *Networks*, 19(6):621–636, 1989.

[4] Palisade Corporation. Guide to evolver: the genetic algorithm solver for microsoft excel. Palisade Corporation, 1998.

[5] G. A. Croes. A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812, 1958.

[6] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Manage. Sci.*, 6(1):80–91, October 1959.

[7] Tobias Dantzig. *Number, the Language of Science*. New York: Free Press, 1954.

[8] A.E. Dooley, W.J. Parker, and H.T. Blair. Modelling of transport costs and logistics for on-farm milk segregation in new zealand dairying. *Computers and Electronics in Agriculture*, 48(2):75–91, 2005.

[9] Merrill M. Flood. The traveling-salesman problem. *Operations Research*, 4(1):61–75, 1956.

[10] Food and Agriculture Organization. The dairy gateway is a single access point for a wide range of information related to dairy production and products. http://www.dinta.cl/wp-content/uploads/2021/03/RSA-DECRETO_977_96_act-02-02-2021.pdf. Accessed: 2021-06-20.

[11] Stefan Irnich, Paolo Toth, and Daniele Vigo. *Chapter 1: The Family of Vehicle Routing Problems*, chapter 1, pages 1–33. 2014.

[12] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[13] S. P. Lloyd. Least squares quantization in pcm. *IEEE Trans. Inf. Theory*, 28:129–136, 1982.

[14] Elizabeth Montero, Dario Canales, Germán Paredes-Belmar, and Raúl Soto. A prize collecting problem applied to a real milk collection problem in chile. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 1415–1422, 2019.

[15] Matías Moreno. Diseño de una técnica metaheurística para el problema de recolección de leche con mezcla y selección. unpublished, 2021.

[16] Germán Paredes-Belmar, Vladimir Marianov, Andrés Bronfman, Carlos Obreque, and Armin Lüer-Villagra. A milk collection problem with blending. *Transportation Research Part E: Logistics and Transportation Review*, 94:26–43, 2016.

[17] Ribeiro Resende, Mauricio G. C. and Celso C. *Greedy Randomized Adaptive Search Procedures*, pages 219–249. Springer US, Boston, MA, 2003.

[18] M. Riff and Elizabeth Montero. A new algorithm for reducing metaheuristic design effort. *2013 IEEE Congress on Evolutionary Computation*, pages 3283–3290, 2013.

[19] Constanza Soto. Un acercamiento meta-heurístico para el problema de recolección de leche con selección y mezcla. unpublished, 2019.

[20] Jorge Villagrán, Elizabeth Montero, and Germán Paredes-Belmar. An iterated local search approach to solve the milk collection problem with blending. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2020.