UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

Repositorio Digital USM **https://repositorio.usm.cl**

Tesis USM TESIS de Pregrado de acceso ABIERTO

2019-10

UN ACERCAMIENTO META-HEURÍSTICO PARA EL PROBLEMA DE RECOLECCIÓN DE LECHE CON SELECCIÓN Y MEZCLA

SOTO CAVIEDES, CONSTANZA ANDREA

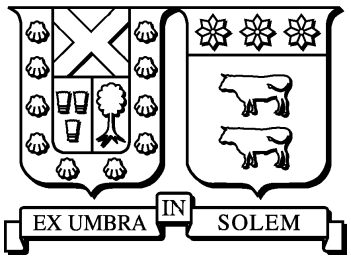
https://hdl.handle.net/11673/49451

Repositorio Digital USM, UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE INFORMÁTICA

SANTIAGO – CHILE



"UN ACERCAMIENTO META-HEURÍSTICO PARA EL PROBLEMA DE RECOLECCIÓN DE LECHE CON SELECCIÓN Y MEZCLA"

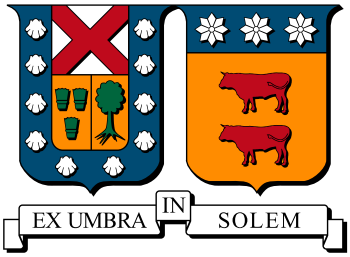
CONSTANZA ANDREA SOTO CAVIEDES

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INFORMÁTICO

PROFESOR GUÍA: ELIZABETH MONTERO URETA

OCTUBRE 2019

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO – CHILE



"UN ACERCAMIENTO META-HEURÍSTICO PARA EL PROBLEMA DE RECOLECCIÓN DE LECHE CON SELECCIÓN Y MEZCLA"

CONSTANZA ANDREA SOTO CAVIEDES

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL INFORMÁTICO

PROFESOR GUÍA: ELIZABETH MONTERO URETA

PROFESOR CORREFERENTE: ERIKA ROSAS OLIVOS

OCTUBRE 2019

MATERIAL DE REFERENCIA, SU USO NO INVOLUCRA RESPONSABILIDAD DEL AUTOR O DE LA INSTITUCIÓN

Agradecimientos

Agradezco a mi familia, tanto a mis padres como a mi tía Ana María y mi abuelo Arturo, por brindarme su apoyo incondicional tanto en el ámbito académico como en la vida en general. Por darme una educación sin distinción de genero ayudándome a cumplir las metas que me propusiese.

Agradezco a mis amigos, quienes son una segunda familia y con quienes en conjunto superamos la etapa universitaria.

También agradezco a la profesora Elizabeth Montero, quien como profesor guía me brindó ayuda y motivación para realizar esta memoria. Además agradezco su alegría y carisma que me ayudaron a iterar sobre este trabajo para obtener buenos resultados.

Finalmente agradezco a Unholster y a todas las personas que de una u otra manera me apoyaron e hicieron más ameno este proceso.

Resumen

El primer proceso dentro de la gran mayoría de las industrias involucra la recolección de materia prima. Este proceso es fundamental en la industria láctea, ya que gran parte del valor final del producto depende de los costos asociados a este proceso.

En esta memoria se aborda el problema de recolección de leches considerando la selección de granjas a visitar y la posibilidad de mezclar calidades de leche que enfrenta una empresa de productos lácteos al sur de Chile. La mezcla se puede realizar tanto dentro de los camiones, como en la planta procesadora, la cual exige una cuota de demandas mínimas. El objetivo de este problema consiste en seleccionar que nodos visitar y armar las rutas óptimas para hacerlo, con el fin de maximizar las ganancias de la empresa.

Para solucionar este problema, se propone un método basado en la meta-heurística Simulated Annealing con dos fases de exploración y dos de intencificación.

Los resultados computacionales obtenidos resultaron ser eficaces para algunas instancias pequeñas, encontrando resultados con una diferencia menor al 5 % con respecto a los resultados obtenidos a partir del modelo de programación entera. Además, para las instancias pequeñas de 40 nodos, demostró tener un mejor desempeño cuando la diferencia entre la cantidad demandada y la capacidad total de la flota es menor. Este comportamiento también se aprecia con la instancia mayor que contiene 500 nodos. Por el contrario, en las instancias grandes se logran mejores resultados cuando no se tienen demandas.

Índice de Contenidos

Agradecimientos III

Resumen IV

Índice de Contenidos V

Lista de Tablas VIII

Lista de Figuras X

Glosario XI

Introducción 1

1. Definición del Problema 4

1.1. Formulación matemática 8

1.1.1. Parámetros 8

1.1.2. Variables 9

1.1.3. Función objetivo 10

1.1.4. Restricciones 11

1.2. Resumen 12

2. Estado del Arte **14**

2.1. Problema de enrutamiento de vehículos - VRP 15

2.2. Variantes del VRP 16

2.3. Múltiples productos - MPVRP 16

2.4. MPVRP con un compartimento 17

2.5. Múltiples compartimentos - MCVRP 18

2.6. Ubicación de puntos de acopio - VRPPD 20

2.7. Múltiples periodos - VRPTW 21

2.8. Simulated Annealing en VRP 21

2.9. VRP en la industria láctea 22

2.10. Recolección de leche con mezcla 23

2.11. Resumen 25

3. Propuesta **26**

3.1. Simulated Annealing 26

3.2. Simulated Annealing para el problema de recolección de leche con mezclas y selección 29

3.2.1. Representación 29

3.2.2. Solución 30

3.2.3. Función de evaluación 30

3.2.4. Factibilidad 31

3.2.5. Estructura 33

3.2.6. Pre-procesamiento 34

3.2.7. Construcción 34

3.2.8. Post-procesamiento 38

3.3. Resumen 47

4. Implementación **48**

4.1. Experimentos 48

4.1.1. Objetivos 49

4.1.2. Instancias de Prueba 49

4.1.3. Sintonización 58

4.1.4. Entorno experimental 62

4.2. Resultados 63

4.2.1. Análisis de componentes 63

4.2.2. Conclusiones del análisis de componentes 79

4.2.3. Comparación con acercamiento completo 80

4.2.4. Evaluación con instancias reales 83

Conclusiones **89**

Anexos **92**

Bibliografía **95**

Índice de tablas

3.1. Producción de cada nodo según calidad de leche. 32

4.1. Características de los nodos con producción atípica. 51

4.2. Características de las instancias pequeñas. 52

4.3. Producciones totales de cada calidad de leche por conjunto. 54

4.4. Características de las instancias reales. 58

4.5. Parámetros y valores considerados en el proceso de sintonización. 59

4.6. Demandas ficticias instancias 3 y 4 con $PD = [0{,}6; 0{,}7; 0{,}8]$. 74

4.7. Calidad de la propuesta versus modelo de programación lineal entera. . . . 80

4.8. Resultados obtenidos y tiempos del conjunto de instancias grandes 87

4.9. Datos del ejemplo utilizado. 92

4.10. Calidad y cantidad de leche producida por granja. 93

4.11. Matriz de distancias de las instancias utilizadas. 94

Índice de figuras

1.1. Ejemplo de distribución de granjas en Chile. Se considera un conjunto de 40 granjas. 5

1.2. Comparación de rutas posibles (b) sin considerar la opción de mezclar leches en camión y (c) permitiendo la mezcla de leche en camión. 7

2.1. (a) División de nodos tipo árbol. (b) Separación de nodos considerando regiones polares 15

2.2. Camión con tres compartimentos, cada uno con su propia capacidad. 19

2.3. División de granjas en zonas y ubicación de posibles puntos de recolección . 24

3.1. Ruleta sesgada para la lista candidata de n_7. 37

3.2. Cálculo de $\Delta_{f.e.}$ para la eliminación de n_6. 40

3.3. Cálculo de $\Delta_{f.e.}$ al agregar n_5. 43

3.4. Candidatos para reordenar n_7 en la ruta del camión k_1. 45

4.1. Componentes principales de un boxplot. 50

4.2. Producciones de nodos según la calidad de leche para las instancias pequeñas. 51

4.3. Litros producidos por cada nodo según el tipo de leche para cada conjunto. 56

4.4. Comparación de los tiempos promedio de ejecución del algoritmo variando el parámetro β. 66

4.5. Calidad de soluciones variando el parámetro β. 67

4.6. Comparación calidad de soluciones variando el parámetro γ. 69

4.7. Tiempos promedio de ejecución del algoritmo variando el parámetro γ. . . 70

4.8. Calidad de soluciones variando el parámetro ε. 72

4.9. Tiempos promedio de ejecución del algoritmo variando el parámetro ϵ. . . . 73

4.10. Tiempos promedio de ejecución del algoritmo variando el parámetro PD. . 75

4.11. Calidad de soluciones variando el parámetro PD. 76

4.12. Calidad de soluciones variando el parámetro τ. 78

4.13. Tiempos promedio de ejecución del algoritmo variando el parámetro τ. . . . 79

4.14. Tiempos de ejecución de cada instancia pequeña. 82

4.15. Resultados obtenidos en las instancias con demandas contra las instancias sin demandas. 84

4.16. Tiempos de ejecución obtenidos en las instancias con demandas versus sin demandas. 86

Glosario

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

SA: Recocido simulado, Simulated Annealing

VRP: Problema de enrutamiento de vehículos, Vehicle Routing Problem

TSP: Problema del vendedor viajero, Traveler Salesman Problem

MPVRP: Problema de enrutamiento de vehículos multi-producto, Multi Product Vehicle Routing Problem

MCVRP: Problema de enrutamiento de vehículos multi-compartimento Multi-Compartment Vehicle Routing Problem

FPLQ: Federación de Productores de Leche de Quebec

VRPPD: Problema de enrutamiento de vehículos con puntos de acopio, Vehicle Routing Problem with Pick-up and Delivery

VRPTW: Problema de enrutamiento de vehículos con ventanas de tiempo, Vehicle Routing Problem with Time Windows

ALNS: Algoritmo adaptativo de gran vecindario, Adaptative Large Neighborhood Search Algoritm

CARS: Sistema de ruteo asistido por computador, Computer Aided Routing System

Introducción

Los procesos de recolección de materia prima están presenten en todas las industrias. En la industria láctea, la logística detrás de la recolección de la leche juega un rol muy importante. La Organización de las Naciones Unidas para la Alimentación y Agricultura (FAO, Foods and Agriculture Organization [1]) señala que más de un 30 % del costo total de la leche corresponde al costo de transporte.

Existen grandes empresas de productos lácteos establecidas en Chile que compran leche a diferentes productores, los cuales se encuentran distribuidos en una amplia zona geográfica. Es más, existen pequeños productores de leche que se agrupan en cooperativas para obtener mejores acuerdos comerciales con un comprador. Lo anterior plantea un gran desafío para el comprador, quien debe visitar más granjas, las cuales no necesariamente producen una gran cantidad de leche. Por otro lado, la leche producida por cada granja posee diferentes calidades, utilizadas para la fabricación de diferentes productos.

Las características especiales de la leche obligan al transporte a poseer ciertas particularidades. La temperatura ideal de la leche, previo al transporte es de 4°C, una vez dentro del camión, esta no deberá superar los 10°C. Existen regiones donde el traslado de la leche se hace en la noche para evitar el calor. La recolección se puede realizar utilizando camiones que poseen compartimentos separados para cada tipo de leche o camiones que recolectan sólo un tipo de leche. La primera opción requiere contar con maquinaria especializada lo cual es costoso. La segunda opción implica recorrer mayores distancias debido a la distribución de las diferentes calidades de leche, perdiendo la oportunidad de visitar granjas que se encuentren en la ruta pero que poseen distinta calidad.

Existe un tercer enfoque, que consiste en permitir mezclar calidades de leches dentro del camión y en la planta. Países como Estados Unidos [7], Panamá [27] y México [19] poseen regulaciones para mezclar las leches. Esto es posible solo si la clasificación final de la leche corresponde a la calidad más baja dentro de la combinación. Aún cuando se degrada la calidad de la leche recolectada, los ahorros en los costos de transporte producen un aumento en las ganancias.

La mezcla de diferentes tipos de productos puede ser aplicada a varias industrias, por ejemplo, Bing et al. [4] resolvió el problema de recolección de reciclaje, donde los camiones recolectores tienen la opción de recoger reciclaje ya clasificado o clasificarlo en la planta de procesamiento. El problema de reciclaje se modela como un problema de enrutamiento de vehículo y se emplea una heurística de búsqueda tabú para mejorar las rutas. Las instancias utilizadas corresponden a datos de casos reales de un municipio holandés.

Este documento presenta un acercamiento para el problema de recolección de leche que enfrenta una empresa de productos lácteos al sur de Chile utilizando el enfoque de mezclas de leche. El objetivo del problema de recolección de mezclas de leche o MCPSB (Milk Collection Problem with Selection and Blending por sus siglas en inglés) es maximizar las ganancias obtenidas considerando el costo de transporte y el beneficio por litro de leche sin romper las restricciones y satisfaciendo la demanda de la planta. Se propone una optimización del proceso de recolección de leche y de la selección de las granjas a visitar, lo cual es de relevancia practica pues solo ha sido abordado previamente por Paredes-Belmar et al. [29]. En comparación con lo realizado anteriormente, este estudio busca resolver el problema mediante un algoritmo de búsqueda incompleta, para obtener buenos resultados en un menor tiempo. Por el contrario, el trabajo realizado por Paredes-Belmar corresponde a una búsqueda completa, con lo cual se obtiene el óptimo global del problema, pero solo es aplicable a instancias pequeñas del problema.

En este trabajo se utiliza un acercamiento meta-heurístico empleando la técnica de recocido simulado (SA, Simulated Annealing por sus siglas en inglés), el cual permite acercarse a un óptimo global en un gran espacio de búsqueda. De esta manera, como resultado, se obtiene para cada camión, perteneciente a una flota heterogénea de vehículos, las granjas que éste debe visitar y en qué orden debe realizar dichas visitas.

Este documento se organiza de la siguiente manera. En el capítulo 1 se define el problema con mayor detalle, presentando su formulación matemática y explicando en detalle todas las restricciones que el problema involucra. Luego, en el capítulo 2, se presenta el estado del arte comenzando con una revisión del problema general de enrutamiento de vehículo (VRP), para luego describir las variantes de recolección de múltiples productos, recolección con múltiples compartimentos y variantes del problema de recolección de leche. A continuación, en el capítulo 3, se describe la propuesta de solución propuesta en este trabajo de memoria. Se presentan una estrategia de búsqueda local basada en Simulated Annealing. Se presenta la propuesta general y los principales componentes especialmente diseñados para el problema a resolver. Posteriormente, en el capítulo 4, se expone el montaje experimental utilizado para evaluar la calidad de la propuesta. Se presentan las instancias de prueba, el entorno de experimentación y las métricas utilizadas para evaluar el algoritmo. Además se presentan los objetivos del proceso de experimentación, que en este caso, están enfocados en evaluar la importancia de los componentes del algoritmo y la comparación con los resultados obtenidos utilizando técnicas completas en instancias pequeñas. Se muestran los resultados y se concluye respecto a cada uno de los objetivos de los experimentos en las instancias de prueba utilizadas. Finalmente en el capítulo 5 se presentan las conclusiones generales del trabajo de memoria realizado y posibles líneas de trabajo futuro que se desprenden de lo realizado.

Capítulo 1

Definición del Problema

El problema de recolección de leche con selección y mezclas o MCPSB (Milk Collection Problem with Selection and Blending) se caracteriza por ser un problema de optimización combinatoria que ocurre cuando se necesita recolectar leche de distintas calidades, desde sus granjas productoras repartidas en una amplia zona geográfica, utilizando una flota de camiones que permiten transportar toda la leche en un solo contenedor. Su objetivo es obtener la ruta óptima para cada camión disponible, indicando cuáles granjas serán visitadas y en qué orden se realizará cada uno de los recorridos. Esto con el fin de maximizar las ganancias al realizar recorridos más cortos al permitir mezclas en rutas, visitando las granjas que sea necesario para al menos suplir las demandas y satisfacer todas las restricciones.

Para la versión del MCPSB de esta memoria se cuenta con un conjunto de camiones, un conjunto de calidades de leche y un conjunto de nodos los cuales contabilizan tanto a las granjas productoras como a la planta de procesamiento. La planta demanda una cierta cantidad mínima a suplir de cada tipo de leche. Es posible sobreabastecer la planta siempre y cuando su recolección genere un beneficio para la empresa.

Por otro lado, el conjunto de nodos se encuentran en una ubicación geográfica determinada, como el ejemplo que se aprecia en la figura 1.1. Esta información viene especificada en una matriz de distancias, la cual además indica que existe un camino entre cada par de nodos, por lo tanto se tiene un grafo conexo. Cada granja produce leche de una sola calidad y se

considera que si un camión visita una determinada granja, éste debe recoger la totalidad de la leche que la granja produzca, restringiendo visitar cada granja a lo más una vez. No es necesario visitar todas las granjas, solo las necesarias para al menos suplir la demanda de cada tipo de leche, por lo tanto, la resolución del problema implica selección de granjas y ruteo.

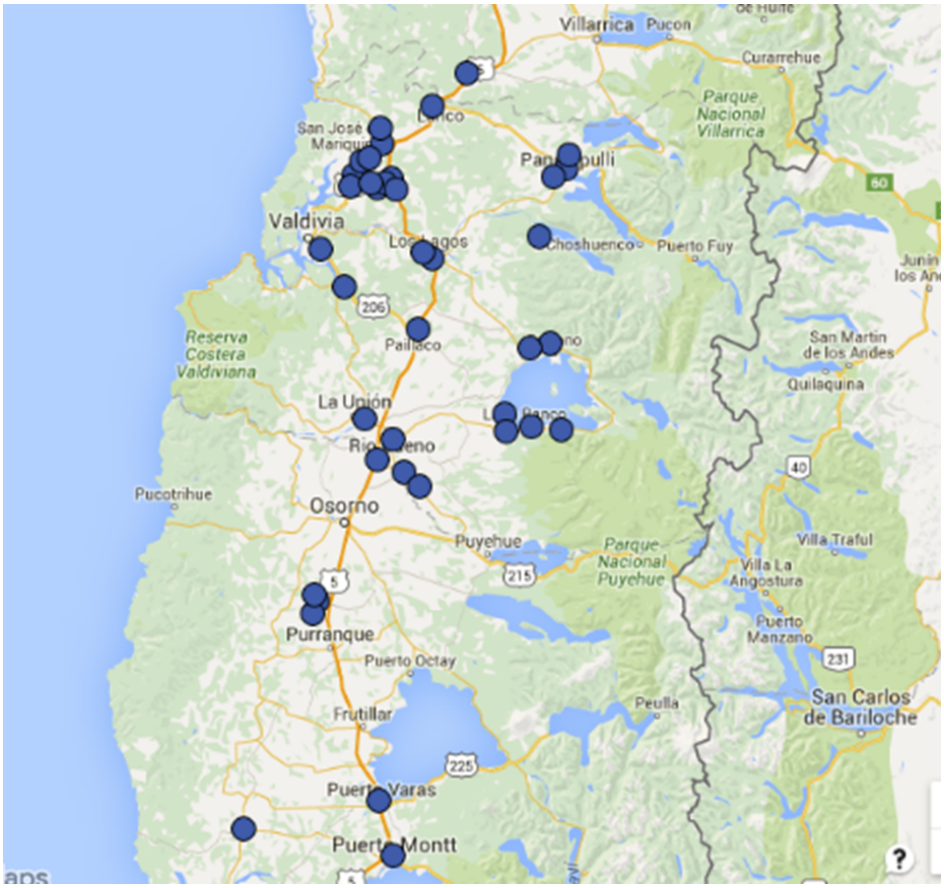


Figura 1.1: Ejemplo de distribución de granjas en Chile. Se considera un conjunto de 40 granjas.
Fuente: Elaboración propia.

Se cuenta con una flota determinada y homogénea de camiones, es decir, se conoce la cantidad de camiones disponibles y todos poseen la misma capacidad de carga, lo cual facilita la selección del camión a utilizar pues todos son iguales. Para que un camión pueda visitar una granja, éste debe tener una capacidad disponible igual o mayor a la producción del nodo, dado que debe recolectar la totalidad de leche que éste produzca. En cada camión es posible

recolectar cualquier tipo de leche, pero la clasificación final de la leche corresponderá a la calidad más baja dentro de la combinación. Esto se conoce usualmente como mezcla en ruta.

Se poseen tres tipos de leches, con calidades decrecientes: la calidad A es mejor que la calidad B y estas dos son mejores que la leche de calidad C. Por lo tanto, al mezclar leche de tipo A con B se obtendrá leche de calidad B, y la combinación de cualquier tipo de leche con leche de calidad C generará leche de calidad C. Además, de ser necesario, se cuenta con la opción de mezclar calidades de leche en la planta para satisfacer las demandas. Esto se conoce como mezcla en planta.

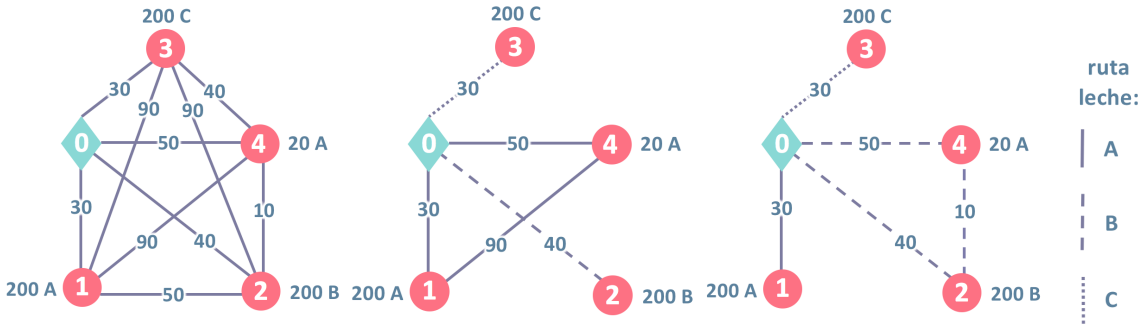
También se asume que todos los camiones parten y terminan su ruta en la planta. Una ruta es definida como el conjunto continuo de viajes entre dos nodos. Cada viaje se compone de un nodo inicial y uno final. Se asume que cada camión puede realizar a lo más una ruta y se debe utilizar la totalidad de la flota.

En consecuencia, el problema consiste en trazar la ruta de los camiones de manera que se logre suplir la demanda de la planta. Después de haber recolectado las leches y realizado las mezclas pertinentes en la planta, se obtiene el volumen final de cada calidad de leche, el cual es multiplicado por su precio de venta. A dicho valor se le resta los costos de transporte obteniendo el beneficio final para la empresa. El objetivo del problema planteado es maximizar este beneficio.

A continuación, en la figura 1.2 se muestra un ejemplo de cómo cambia una solución cuando se permite la mezcla de calidades de leche dentro del camión en comparación a utilizar camiones dedicados a un solo tipo de leche. El conjunto de nodos se ha representado como nodos del grafo y se han numerado desde 0 a 4. El nodo 0, en verde, corresponde a la planta de procesamiento. La granja 1 produce $200L$ de leche A; la granja 2 produce $200L$ de leche B; la granja 3 produce $200L$ de leche C y la granja 4 produce $20L$ de leche A, estos valores se encuentran acompañando a cada granja en la figura 1.2a. El valor en cada arco corresponde a las unidad de distancia recorrida y éste no posee relación con el largo ilustrado. El costo de transporte por unidad de distancia recorrida corresponde a $1{,}0u.m.$. Las ganancias por litro de leche corresponden a $1{,}0u.m.$ para la calidad A, $0{,}7u.m.$ para B y $0{,}3u.m.$ para C. La demanda de la planta corresponde a $200L$ de leche para cada calidad, es decir, se debe recolectar $200L$

de leche de calidad A, 200L de leche de calidad B y 200L de leche de calidad C. Se posee una flota de 3 camiones con capacidad de 220L cada uno.

En la figura 1.2b se ilustran posibles recorridos de los camiones cuando no está permitido mezclar diferentes tipos de leche dentro de los estanques. El costo total de transporte corresponde en este caso corresponde a 310$u.m.$, obtenido al realizar las rutas $0-1-4-0$ (leche tipo A, recorriendo 170$u.d.$), $0-2-0$ (leche tipo B, recorriendo 80$u.d.$) y $0-3-0$ (leche tipo C, recorriendo 60$u.d.$) y tomando en cuenta un costo de transporte por unidad de distancia recorrida igual a 1,0$u.m.$. La ganancia corresponde a 420$u.m.$, obtenido al multiplicar los 220L de leche tipo A, 200L de leche tipo B y 200L de leche tipo C con sus respectivas ganancias por litro. Con lo cual se obtiene un beneficio final de 110$u.m.$ al realizar la resta entre la ganancia total generada por la leche y el costo de transporte.



(a) Distribución de nodos. (b) Camiones dedicados. (c) Camiones con mezclas.

Figura 1.2: Comparación de rutas posibles (b) sin considerar la opción de mezclar leches en camión y (c) permitiendo la mezcla de leche en camión.
Fuente: Elaboración propia

Por otro lado, en la figura 1.2c se ilustran posibles recorridos de los camiones considerando la opción de mezclas dentro de los estanques. Los recorridos corresponden las rutas $0-1-0$ (leche tipo A, recorriendo 60$u.d.$), $0-2-4-0$ (leche tipo B, recorriendo 100$u.d.$) y $0-3-0$ (leche tipo C, recorriendo 60$u.d.$). El costo de transporte corresponde a 220$u.m.$ y la ganancia a 414$u.m.$, la cual se obtiene al multiplicar los 200L de leche tipo A, los 220L de leche tipo B y los 200L de leche tipo C recolectados, con su respectivas ganancias por litro. Con esto se genera un beneficio final de 194$u.m.$.

En este ejemplo, al comparar las ganancias obtenidas por ambas modalidades, es posible observar que al tener camiones dedicados se obtiene un beneficio mayor en $6u.m.$, pero el costo de transporte es $110u.m.$ menor al permitir mezcla. Con estas diferencias de costos y ganancias, al permitir la mezcla de tipos de leche se obtiene un resultado 76 % más alto que al tener camiones dedicados.

En el siguiente apartado se presenta la formulación matemática la cual describe el problema de recolección de leche con selección y mezclas.

1.1. Formulación matemática

El modelo matemático propuesto para solucionar el MCPSB se basa en el postulado por Paredes-Belmar et al.en [29]. Primero se definen los parámetros asociados al problema descrito. A continuación se presentan las variables de decisión de la formulación. Luego se presenta la función objetivo utilizada para medir la calidad de las soluciones del problema. Finalmente, se presentan y describen todas las restricciones que se consideran para la versión del problema de recolección de leche con selección y mezclas estudiada en este trabajo de memoria.

1.1.1. Parámetros

A continuación se describen los parámetros necesarios para definir una instancia del problema de recolección de leche con selección y mezclas.

$G(N, A)$: G un grafo completo. Es decir, se considera la existencia de un camino entre cada par de nodos.

N : Conjunto de nodos, comprende a granjas productoras de leche y a la planta procesadora.

A : Conjunto de caminos entre las granjas. Existe un camino entre cada par de granjas.

n_0 : Nodo correspondiente a la planta procesadora.

A^0 : Conjunto de arcos que conectan la planta procesadora con cada una de las granjas

K : Conjunto de camiones de recolección.

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 un solo tipo de leche.

Q^k : Capacidad del camión $k, \forall k \in K$. En este caso, se considera una flota de camiones homogénea, es decir, 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.

P^t : Demanda por parte de la planta procesadora de leche de calidad $t, \forall t \in T$.

1.1.2. Variables

A continuación se presentan las variables de decisión del problema a resolver. Se consideran tres conjuntos de variables binarias asociadas a las secuencias de visitas de granjas, los tipos de leche recolectados en las granjas y los tipos de leche entregados en la planta procesadora para cada uno de los camiones. Además, se consideran dos variables enteras, asociadas a la cantidad de leche entregada en la planta por cada uno de los camiones y cantidad de leche final después de la mezcla en planta.

$$x_{ij}^{k} \begin{cases} 1, & \text{si el camión } k \text{ viaja del nodo } i \text{ al } j, \forall k \in K, \forall (i,j) \in A \\ 0, & \text{si no} \end{cases}$$

$$y_{i}^{kt} \begin{cases} 1, & \text{si el camión } k \text{ carga leche de calidad } t \text{ en la granja } i, \forall k \in K, \forall t \in T, \forall i \in N \\ 0, & \text{si no} \end{cases}$$

$$z^{kt} \begin{cases} 1, & \text{si el camión } k \text{ entrega leche de calidad } t \text{ a la planta, } \forall k \in K, \forall t \in T \\ 0, & \text{si no} \end{cases}$$

w^{kt} : Volumen de leche de calidad t que el camión entrega en la planta, $\forall t \in T \forall k \in K$.

v^{tr} : Volumen de leche de calidad t entregado a la planta, mezclado para usarlo como leche de calidad $r, \forall t, r \in T$.

Se considera entonces que el dominio de las variables corresponde a:

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

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

1.1.3. Función objetivo

Como se ha mencionado anteriormente, el objetivo de este problema es maximizar la ganancia final, por lo tanto la función objetivo corresponde a:

$$fo = \text{máx} \sum_{r \in T} \sum_{t \in T} L^{r} v^{tr} - C \sum_{k \in K} \sum_{(i,j) \in A} d_{ij} x_{ij}^{k}$$

Es decir, la diferencia entre el total del valor de venta de cada tipo de leche considerando el volumen final de cada tipo de leche posterior a la mezcla en la planta ponderado por la

ganancia asociada a cada tipo de leche y la distancia total recorrida por todos los camiones ponderada por el costo de cada kilómetro.

1.1.4. Restricciones

Las restricciones se clasifican en dos grupos, primero se presentan las restricciones asociadas al proceso de recolección y luego las restricciones relacionadas con la mezcla de diferentes tipos de leche.

Restricciones de recolección

1. La cantidad de leche recolectada por cada camión k no debe superar su capacidad.

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

2. No todas las granjas deben ser visitadas. Cada granja puede ser visitada por a lo más un camión. Esta restricción permite seleccionar el conjunto de granjas a visitar.

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

3. Todos los camiones deben tener una ruta partiendo en la planta. Esta restricción se encarga de controlar que todas las rutas construidas comiencen en la planta de procesamiento.

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

4. Recorrido continuo de los camiones. Esta ruta permite controlar el flujo de visitas de los camiones en sus respectiva rutas.

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

5. Si un camión carga leche en cierta granja, ésta debe encontrarse en la ruta de dicho camión.

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

6. Como mínimo se debe suplir las demandas de la planta. Esta restricción se encarga de suplir la demanda de cada uno de los tipos de leche. Considera la leche total recolectada posterior al proceso de mezcla en planta.

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

7. La leche que llega a la planta es toda la contenida en el camión. Toda la leche recolectada se entrega en la planta procesadora.

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

Restricciones de mezclas

8. Balance de la cantidad de leche de cada tipo que llega a la planta y la cantidad de leche de cada tipo después de realizar la mezcla en la planta.

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

9. Controla el tipo de leche recolectada por cada camión de acuerdo a los nodos visitados en su ruta.

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

10. La leche entregada por el camión en la planta es de una sola calidad. Esta restricción se encarga de controlar la mezcla en el camión de acuerdo a los nodos visitados.

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

1.2. Resumen

En este capítulo se describe el problema de recolección de leche con selección y mezclas. El problema considera un conjunto de granjas repartidas geográficamente en una región dada y una planta procesadora encargada del proceso de producción asociado a la leche. El problema

se enfoca en la minimización de los costos de recolección y la maximización de la ganancia asociada a las diferentes calidades de leche recolectadas. Para la recolección de la leche se cuenta con una flota heterogénea de camiones que poseen un solo compartimento para transportar la leche. Dado que los camiones poseen un único compartimento, se permite la mezcla de diferentes calidades de leche. Se considera también la posibilidad de realizar mezcla en planta para satisfacer la demanda de leche de cada tipo realizada por parte de la planta. En el problema tratado no es necesario visitar todas las granjas, por lo tanto se puede entender como la unión de dos problemas, uno de selección y uno de ruteo asociado a la selección.

Capítulo 2

Estado del Arte

El problema de recolección de leche con selección y mezcla (MCBSP por sus siglas en inglés) se puede considerar como una combinación entre un Problema de ruteo de vehículos, más conocido como VRP (Vehicle Routing Problem [9]) y el problema de la mochila (Knapsack Problem [10]). El primero busca responder la pregunta "¿Cuál es el conjunto óptimo de rutas para una flota de vehículos que debe satisfacer las demandas de un conjunto dado de clientes?" y el segundo busca encontrar el conjunto óptimo de objetos a cargar en una mochila sin sobrepasar su capacidad. Por lo tanto, puede considerarse que el problema de la mochila resuelve el problema de la selección de granjas a visitar y el problema de ruteo decide en qué orden realizarlo.

Esta sección se enfoca en la literatura relacionada con problemas de ruteo especialmente aquellos basados en la industria láctea, resaltando cómo éstos se asemejan y en qué se diferencian con el problema estudiado. Para una clasificación más exhaustiva de los VRPs existentes, se puede consultar el trabajo realizado por Raff [30]. La sección comienza presentando el VRP clásico y los primeros trabajos donde se aborda la recolección de leche como un VRP particular. A continuación se exponen diferentes variantes del VRP, entre los cuales se consideran: recolección de múltiples productos, vehículos con múltiples compartimentos, ubicación de puntos de acopio y múltiples periodos de recolección. Para finalizar se exponen problemas de ruteo que han sido abordados con la meta-heurística Simulated Annealing y que presentan más elementos en común con el acercamiento propuesto en este trabajo.

2.1. Problema de enrutamiento de vehículos - VRP

El primer trabajo que aborda el VRP es el realizado por Dantzig y Ramser [9] en 1959 basado en un problema de distribución de combustible donde se concluye que este problema es de clase NP-Completo. Esta conclusión se debe a que el VRP puede ser transformado en el clásico problema del vendedor viajero, siendo este último de clase NP-Completo. El problema del vendedor viajero, o TSP por sus siglas en inglés, busca responder a la pregunta: dada una lista de ciudades y las distancias entre cada par de ellas, ¿Cuál es la ruta más corta posible que visita cada ciudad exactamente una vez y al finalizar regresa a la ciudad origen?. Por lo tanto, al considerar un solo vehículo que no posee restricción de carga, el VRP se transforma en un TSP. Formalmente el VRP fue descrito en 1983 en [5] como: "un conjunto de vehículos ubicados en un depósito central que son utilizados para visitar clientes localizados geográficamente dispersos con el objetivo de satisfacer las demandas (conocidas) de los clientes". El VRP exige que cada cliente sea visitado exactamente una vez por uno de los vehículos, respetando las restricciones de capacidad, el tiempo máximo permitido de trabajo, distancia máxima recorrida, entre otros.

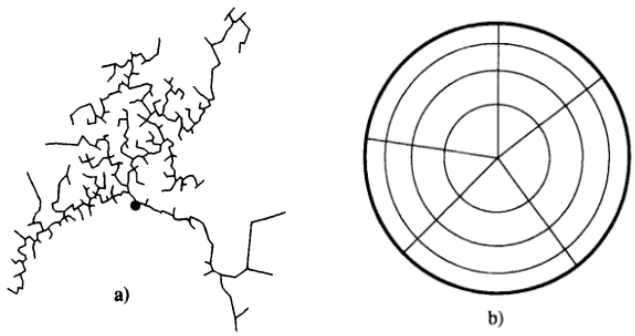


Figura 2.1: (a) División de nodos tipo árbol. (b) Separación de nodos considerando regiones polares

Fuente: "Parallel iterative search methods for vehicle routing problems" [33]

Según una investigación realizada por Laporte et. al. [23] los mejores resultados obtenidos para solucionar el VRP son los algoritmos basados en búsqueda tabú, siendo la implementación de Taillard [33] la que encontraba mejores resultados a la fecha del estudio. En ella, identifica la existencia de dos tipos de distribución de nodos, una euclidiana que se considera

más uniforme y otra no euclidiana. Para resolver el problema propone separar los nodos en regiones las cuales se conforman según el tipo de distribución de los nodos y luego resuelve cada sector independientemente. Cuando la distribución es uniforme, propone utilizar regiones según coordenadas polares como se ilustra en la figura 2.1.b. Por otro lado, cuando la distribución no es uniforme propone utilizar un árbol para cubrir los nodos y luego realizar regiones por ramas, como se ve en la figura 2.1.a.

2.2. Variantes del VRP

A lo largo del tiempo, se han estudiado diversas variantes del VRP en las cuales se agregan diferentes restricciones para poder acercar el problema a la realidad. A continuación se presentan las modificaciones más típicas del VRP desde el punto de vista del problema de recolección de leche.

2.3. Múltiples productos - MPVRP

En esta sección se describen acercamientos a investigaciones que estudian versiones del MPVRP vehículos de carga con un solo espacio de almacenamiento.

Una primera variante, que acerca el VRP al MCBSP, es la consideración de diferentes calidades de leche, es decir, múltiples productos. Este acercamiento es conocido como el problema de ruteo de vehículos multi-productos (MPVRP, multi-product vehicle routing problem). Los acercamientos MPVRP consideran tres escenarios: trabajar con rutas separadas, uso de compartimentos y posibilidades de mezcla de productos. A continuación se describen algunos acercamiento de recolección multi-productos relacionados a la recolección de leche.

2.4. MPVRP con un compartimento

En [14], Dooley et. al. enfrenta un MPVRP basado en la industria láctea utilizando datos de Nueva Zelanda. Para esto consideraron dos calidades de leche y separaron las granjas en dos grupos de 30 granjas cada uno, los cuales denominaron zona norte y zona sur. En el estudio consideran dos camiones con diferentes capacidades, los cuales recolectan toda la leche de un solo tipo, por lo que se asigna el tipo de leche más producida al camión con mayor capacidad. En el problema no se posee una demanda de leche, se debe recolectar toda la leche. Las instancias de prueba fueron generadas tras modelar cambios porcentuales en la producción de leche de cada una de las granjas. Con la utilización de un algoritmo genético obtuvieron diferentes rutas para cada zona y realizaron comparaciones entre ellos centrándose principalmente en los costos de transporte asociados a la recolección de la leche y cómo estos varían entre rutas.

Liu et. al. [24] también analizan un MPVRP pero enfocado en un problema de entrega de paquetes. En este estudio, a diferencia del MCBSP, se posee una cantidad fija de clientes a visitar, por lo que no se considera la selección de nodos. Además se poseen múltiples puntos de carga de productos, por lo que se propone la entrega parcial de productos al cliente a medida que el camión cargue los productos solicitados. Se genera una ruta inicial utilizando el algoritmo de Dijkstra, también llamado algoritmo de caminos mínimos y luego se selecciona un nodo al cual no se le entrega la totalidad de su demanda. De esta manera, este nodo se considera como si fuese dos nodos con demandas que suman la inicial y puede ser visitado aun cuando el camión no posea la demanda inicial. Los resultados, basados en 14,000 instancias generadas aleatoriamente, demuestran una mejora notoria obtenida por el algoritmo de búsqueda local que permite llevar diferentes tipos de paquetes sobre una versión que no lo permite.

Junqueira et.al. [20] centran su preocupación en la dimensión de los productos. En su trabajo los autores consideran el reparto de productos embalados, tomando en cuenta que éstos poseen tres dimensiones. El objetivo es encontrar rutas que generen el mínimo costo para un conjunto de vehículos con diferentes capacidades que comienzan y terminan su recorrido en la bodega, visitando a todos los clientes. Este objetivo se ve condicionado por los objetos

que se pueden llevar en cada vehículo ya que, además del ordenamiento de los embalajes dentro del vehículo correctamente, el problema también toma en cuenta la fragilidad de las cajas e incluso restricciones relacionadas con la estabilidad de la carga, situación en la cual puede ocurrir un derrumbe. En esta ocasión realizaron un modelo de programación lineal entera y lo resolvieron utilizando el solver GUROBI con un tiempo máximo de computo de 4 horas. Dada la complejidad del problema, solo se consideraron instancias aleatorias con 4 a 9 nodos.

2.5. Múltiples compartimentos - MCVRP

Los acercamientos de esta sección describen estudios realizados para problemas MPVRP que consideran camiones con compartimentos, evitando la mezcla de productos. Esta versión del VRP es conocida como el problema de ruteo de vehículos multi-compartimentos (MCVRP, multi-compartment vehicle routing problem).

Caramia et. al. [6] resuelven un MCVRP para el problema de una empresa láctea italiana que recolecta cuatro tipos de leche desde 158 granjas utilizando camiones especiales con múltiples tanques. La figura 2.2 muestra la estructura de los camiones de recolección considerados en el estudio. El objetivo de este trabajo es demostrar como las técnicas de investigación de operaciones pueden ayudar en el rendimiento diario de la empresa. Para este caso se utilizaron dos formulaciones matemáticas resueltas con el solver CPLEX 7,0 y búsqueda local en conjunto con el mecanismo de reinicio múltiple. El primer paso minimiza el número de vehículos que serán utilizados y el segundo minimiza la ruta. Las soluciones son comparadas con el proceso utilizado anteriormente por la empresa y demuestran una gran mejora que se ve reflejada en ahorro para la empresa.

El Fallahi et. al. [15] consideran un conjunto de clientes que ordenan diferentes productos y vehículos repartidores que poseen compartimentos dedicados para cada tipo de producto. La cantidad demandada por un cliente de cierto producto debe ser satisfecha solo por un vehículo. Las 20 instancias utilizadas poseen entre 50 y 199 nodos. Las instancias fueron creadas a partir de instancias ya conocidas para el VRP, definidas por los autores Christofides y Eilon

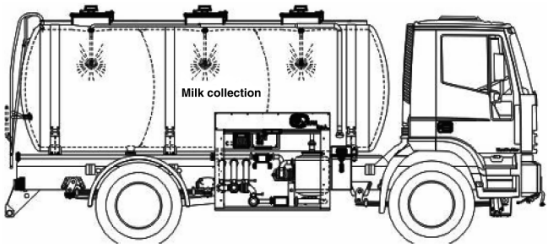


Figura 2.2: Camión con tres compartimentos, cada uno con su propia capacidad.
Fuente: “A Milk Collection Problem with Incompatibility Constraints.” [6]

[8], modificándolas para contemplar la existencia de dos productos. Este problema es resuelto de tres maneras: utilizando una heurística de construcción la cual no realiza iteraciones de mejora, utilizando búsqueda tabú y utilizando un algoritmo memético, el cual es una extensión del tradicional algoritmo genético, con una fase de optimización basada en desvincular y vincular rutas. Los resultados demuestran que búsqueda tabú obtiene resultados levemente mejor que el algoritmo memético, pero requiere más tiempo de computación.

En Canadá también se han realizado estudios sobre recolección de leche utilizando camiones con múltiples compartimentos. El 2015 Lahrichi et al. [22] utilizan diferentes camiones con dos compartimentos para la recolección de tres calidades de leche demandadas por más de una planta. Para la resolución del problema, utilizan una heurística de búsqueda tabú cuando no se ha logrado asignar los camiones a las granjas. Por otro lado, cuando ya se sabe qué granjas visita cada camión se utiliza un heurística GENIUS, propuesta por Gendreau [16], la cual considera un paso de inserción y una post-optimización. En su trabajo, los autores resuelven instancias reales de entre 73 y 226 entregadas por la Federación de Productores de Leche de Quebec (FPLQ, Fédération des Producteurs de Lait du Québec). Su estudio se divide en dos partes, primero estimar el valor que será pagado por hectolitro de leche. Segundo, generar las mejores rutas para recolectar la leche. Sus resultados finales se ven muy afectados por los valores obtenidos en el primer paso, por lo tanto concluyen que es muy importante pactar un buen valor inicial de compra.

Por su parte, Dayarian et. al. [12] , realiza un estudio ese mismo año donde se generan instancias de prueba de manera aleatoria utilizando un máximo de 100 granjas distribuidas en áreas de cierto tamaño y más de una planta de procesamiento. Además, para facilitar la

generación de resultados factibles, la demanda total de la plata siempre corresponde al 90 % del total de leche producida por las granjas. En este estudio se deben visitar todas las granjas, por lo tanto no existe selección de nodos como en el MCPSB. Además se utiliza una flota de camiones no heterogénea, donde cada camión posee diferente cantidad de compartimentos y capacidad. Para resolver el problema utiliza un algoritmo de bifurcación y corte, obteniendo resultados de buena calidad especialmente para las instancias con menos de 50 productores.

2.6. Ubicación de puntos de acopio - VRPPD

Por otro lado, hay estudios donde, además de camiones con múltiples compartimentos, se utilizan puntos de acopio, es decir, lugares donde las granjas van a dejar sus producciones. De esta manera, concentran las producciones en un solo punto el cual es visitado por los camiones. Este acercamiento se conoce como VRPPD (por sus siglas en inglés Vehicle Routing Problem with Pick-up and Deliveries).

Sethanan y Pitakaso [32] utilizan modificaciones del algoritmo de evolución diferencial, perteneciente a la categoría de computación evolutiva, para resolver este problema. Al igual que en la mayoría de las soluciones de VRP, la solución corresponde a un vector con el orden de los nodos a visitar para cada vehículo. En este caso, dicho vector se obtiene tras realizar una decodificación de un vector, llamado *target vector* o vector objetivo en español, el cual toma en consideración otros atributos del problema como el tiempo. En su estudio los autores utilizaron 10 instancias de prueba con un total de hasta 40 granjas a visitar y el tiempo de computación fue de 1 hora. Con las cinco modificaciones que proponen, las cuales consideran un paso de reencarnación y otro de supervivencia, obtienen mejores rutas. Con ello, mejoran los costos totales y utilizan menos camiones, ya que en su acercamiento, a diferencia del MCPSB, se permite no utilizar toda la flota de camiones. Como trabajo futuro, sugieren la mezcla de leche en cada punto de acopio.

2.7. Múltiples periodos - VRPTW

Otra variante del VRP que es común cuando el problema se centra en la industria láctea, considera múltiples periodos de recolección. Esta variante es conocida por sus siglas VRPTW, las cuales provienen de su nombre en inglés: vehicle routing problem with time windows.

Dayarian et al. abordan esta versión del VRPTW dos veces. En ambas ocasiones utiliza datos reales obtenidos en Quebek, Canadá. En su primer trabajo [11] considera dividir la recolección de leche total demandada en 5 periodos de recolección obteniendo buenos resultados solo con instancias de hasta 20 nodos. Para su solución utiliza un algoritmo de branch-and-price con un tiempo máximo de ejecución de 5 horas utilizando Cplex. Al año siguiente, realiza un segundo trabajo [13] donde propone utilizar un algoritmo de búsqueda adaptativo con gran vecindario, ALNS (del inglés Adaptive Large Neighborhood Search Algorithm). En el problema considera un horizonte temporal de varios días o incluso meses en los cuales existen variaciones en la producción láctea total de cada granja. Las soluciones obtenidas son comparadas con su trabajo anterior cuando utiliza instancias de hasta 60 nodos. Para las instancias que consideran entre 100 y 200 granjas, dado que no poseen los valores óptimos, calculan los límites superiores e inferiores de la función objetivo para poder comparar sus resultados. El algoritmo propuesto obtiene resultados de buena calidad con poco esfuerzo computacional para las instancias menores a 60 nodos, en cambio para algunas instancias de más de 100 nodos no se obtiene una solución de buena calidad en las 10 horas de cómputo máximo.

2.8. Simulated Annealing en VRP

Además de existir diferentes variaciones del VRP, existen muchos métodos para abordarlo, utilizando algoritmos de búsqueda completa o incompleta. En los estudios expuestos en las secciones anteriores se ha hablado de la utilización de algoritmos tales como búsqueda tabú, algoritmos genéticos o basados en ellos, algoritmos de bifurcación y corte e incluso la utilización de solvers como CPLEX o GUROBI. Para este estudio, se propone resolver el MCPSB utilizando un algoritmo de recocido simulado, más conocido como Simulated

Annealing o SA por sus siglas en inglés. El SA pertenece a los algoritmos de búsqueda incompleta y se encuentra basado en el proceso de recocido del acero y las cerámicas, técnica que consiste en calentar y luego enfriar lentamente el material para variar sus propiedades físicas. Es por esto que el algoritmo realiza una búsqueda local la cual se ve condicionada estocásticamente por una variable de temperatura. En el capitulo 3 se explica más detalladamente esta meta-heurística y cómo se utiliza en esta investigación. En la literatura, Alfa et. al. [3] utiliza SA para resolver un VRP clásico como el descrito anteriormente en la sección 2.1. En su trabajo, utiliza el principio de armar una gran ruta y luego separarla en rutas más pequeñas. Este principio se conoce como RFCS por sus siglas del ingles Route-first Cluster-second, cuya traducción es *primero la ruta luego el clúster*. Para esto, aplica un algoritmo de búsqueda local conocido como 3-opt modificado. Este algoritmo involucra intercambiar tres viajes entre rutas para explorar el espacio de búsqueda. Para seleccionar los viajes a intercambiar, se utiliza SA siempre tomando en cuenta las restricciones del problema, como por ejemplo la capacidad de los camiones. Las instancias utilizadas poseen más de 30 nodos, dado que buscan evaluar el desempeño del algoritmo en instancias grandes, y presentan ejemplos numéricos para tres instancias de 30, 50 y 75 nodos. El tiempo de cómputo para las dos primeras instancias es de 1 hora, para las cuales logran obtener el mejor resultado encontrado hasta la fecha por otros autores. Por otro lado, para la instancia mayor utilizaron un criterio de término de 2 horas, obteniendo un buen resultado, pero no el mejor. En la investigación se concluye que el desempeño del algoritmo se ve muy condicionado por la solución inicial utilizada, pero que la utilización de SA con otras heurísticas mejora bastante los resultados finales.

2.9. VRP en la industria láctea

Sankaran y Ubgade [31] en 1994 fueron los primeros en abordar la recolección de leche como un caso especial de VRP. Diseñaron rutas de recolección para minimizar los costos de transportes para resolver un caso real de 70 granjas en India. Para ello, desarrollaron un sistema de apoyo a la Decisión (decision support system o DSS por sus siglas en ingles) bautizado como CARS, nombre generado por las siglas del ingles *Computer Aided Routing System* que

se traduce como *sistema de ruteo asistido por computadora*. A partir de sus resultados fueron capaces de obtener un ahorro anual de $15,000USD. En su estudio consideran una flota de camiones con diferentes capacidades. Además, consideran no exceder cierto tiempo entre la recolección y la entrega dependiendo de la época del año, ya que la leche debe mantener cierta temperatura a lo largo de todo el proceso.

2.10. Recolección de leche con mezcla

El problema de recolección de leches utilizado un acercamiento que permite la mezcla de leche ha sido abordado previamente solamente dos veces por Germán Paredes-Belmar et. al. En [29] el 2016 se presenta por primera vez el problema de recolección de leche permitiendo mezcla. Propone una formulación entera mixta para resolver el problema y propone un algoritmo de bifurcación y corte donde utiliza un nuevo corte y otros ya conocidos para resolver de manera óptima las instancias de mediano tamaño. Además, utiliza un procedimiento heurístico para resolver instancias grandes. Para esto, divide la instancia real en conjuntos de granjas por su ubicación y luego utiliza un algoritmo de corte para resolver cada una. Las instancias generadas son de máximo 100 nodos y la instancia real posee 500 granjas. El objetivo del estudio es diseñar una heurística que brinde resultados favorables dentro de los límites de tiempo, el cual no es crítico, pues la ruta no cambia constantemente. La solución demora 1,65 horas en encontrar una solución óptima utilizando un computador de 4 núcleos. Las soluciones implementadas con este enfoque se comparan con las implementadas por la empresa y con las soluciones obtenidas al recolectar la leche sin permitir la mezcla, donde se tiene un problema de enrutamiento de vehículo (VRP) para cada calidad de leche. Además, se considera la utilización de camiones con un solo compartimento o con múltiples compartimentos, con opción de mezclar o no la leche. También se compara la recolección opcional con la obligatoria de la demanda de la planta.

Los resultados demuestran que con las instancias utilizadas, la mezcla de calidades de leche siempre domina sobre la recolección de leche sin mezclar. Por otro lado, en la mayoría de los casos, utilizar camiones con múltiples compartimentos es mejor que utilizar los que poseen un gran tanque, pero siempre predomina la mezcla de leche dentro de ellos. Por otro lado, la

mezcla permite la viabilidad en todas las instancias ya que si no se permite mezclar, asignar de forma incorrecta una calidad de leche a un compartimento lleva a soluciones no factibles. Para la instancia real, utilizar camiones con un solo tanque y permitir la mezcla, es lo que genera mayor ganancia, principalmente por el ahorro en el costo de transporte.

En [28] los autores estudian el problema de ubicación de puntos de acumulación para recolectar la leche de granjas pequeñas y distantes. En este estudio se utiliza la misma aproximación del anterior, un modelo de programación entera que usa un algoritmo de bifurcación y corte para las instancias pequeñas y un procedimiento heurístico que incluye un sistema de colonias de hormigas para manejar las instancias grandes en un tiempo razonable. Se resuelve el mismo caso real de 500 granjas y se consideran 112 posibles puntos de recolección. Dado el tamaño de la instancia y su consecuente complejidad, la red de granjas se dividió en muchas zonas pequeñas como se ilustra en la figura 2.3.

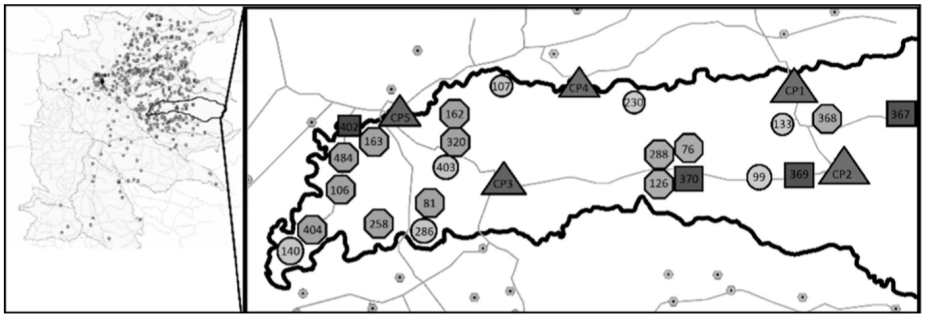


Figura 2.3: División de granjas en zonas y ubicación de posibles puntos de recolección . Fuente: “The milk collection problem with blending and collection points” [28]

Los resultados mostraron que el algoritmo de no fue capaz de resolver el problema completo en un tiempo razonable, aún utilizando divisiones por zonas. Sin embargo la heurística sin división por zonas encontró una solución muy satisfactoria en términos de ganancias en cuestión de minutos. Los hallazgos del estudio demuestran que el uso de puntos de recolección ayuda a reducir los costos de transporte y, por lo tanto, aumenta las ganancias. Si además se permite la mezcla de diferentes calidades de leche, los beneficios financieros son aún mayores.

2.11. Resumen

Como es posible apreciar, a lo largo del tiempo se han planteado diferentes problemas de recolección de leche los cuales se han abordado con algoritmos de búsqueda completa e incompleta. En la gran mayoría de los casos, las instancias poseen menos de 100 nodos o se obtienen buenos resultados solo para instancias pequeñas. En este trabajo se busca abordar instancias de gran tamaño, las cuales son más cercanas a la realidad. Por otro lado, el problema abordado considera camiones con un solo compartimento y en la mayoría de la literatura se consideran camiones con múltiples compartimentos, los cuales son más costosos. Además, el problema planteado en este trabajo, permite la mezcla de calidades de leche y posee restricciones relacionadas a la mezcla obtenida, lo cual ha sido abordado previamente solo dos veces en [28] y [29] por el mismo autor. Finalmente es importante hacer notar que la selección de nodos y el suplir mínimamente la demanda de la planta son otras particularidades del problema que lo diferencian sustancialmente de los que se pueden encontrar en la literatura.

Capítulo 3

Propuesta

En este capítulo, se presenta el acercamiento propuesto en este trabajo de memoria para resolver el problema de recolección de leche con selección y mezcla. El método propuesto es un acercamiento de búsqueda local basado en Recocido Simulado (SA, Simulated Annealing, por sus siglas en ingles). A continuación se describe de manera general el método Simulated Annealing en el cual se basa la propuesta. Luego, se presenta el algoritmo propuesto, explicando primeramente su estructura para luego describir en detalle cada una de sus componentes.

En el anexo 4.9 se encuentran todos los datos utilizados como ejemplo a lo largo del documento.

3.1. Simulated Annealing

También conocida como templado simulado, Simulated Aanneling es una meta-heurística propuesta inicialmente por Scott Kirkpatrick, C. Daniel Gelatt y Mario P. Vecchi en 1983 [21]. El método corresponde a una adaptación del algoritmo Metropolis-Hastings postulado en 1953 [26], el cual corresponde a un método de Montecarlo utilizado para generar muestras de estados de un sistema termodinámico. Simulated Annealing es un algoritmo de búsqueda meta-heurística para problemás de optimización. El objetivo de estos algoritmos

es encontrar una buena aproximación al valor óptimo, conocido como óptimo global, de una función en un espacio de búsqueda grande. Esta técnica se inspira en el proceso de recocido del acero y las cerámicas, el cual consiste en calentar y luego enfriar lentamente el material para variar sus propiedades físicas. Como se describe en [17] el calor causa que los átomos aumenten su energía y así puedan desplazarse de sus posiciones iniciales (un mínimo local de energía), por otro lado, el enfriamiento lento les da mayores probabilidades de recristalizar en configuraciones con menor energía que la inicial (mínimo global).

Algoritmo 1: SIMULATED ANNEALING

```
1  inicializar T = temperaturaInicial
2  inicializar random solucionActual
3  inicializar solucionMejor = solucionActual
4  mientras criterio de término hacer
5      mientras criterio de interrupción hacer
6          solucionNueva = seleccionar un nuevo punto en N(solucionActual)
7          si f(solucionNueva) es mejor que f(solucionActual) entonces
8              solucionActual = solucionNueva
9          fin
10         si random ([0, 1]) < e^(Δ_f.e./T) entonces
11             solucionActual = solucionNueva
12         fin
13         si f(solucionActual) es mejor que f(solucionMejor) entonces
14             solucionMejor = solucionActual
15         fin
16         t = t + 1
17         T = g(T, t)
18     fin
19     T = temperaturaInicial
20 fin
21 devolver solucionMejor;
```

La estructura general de un acercamiento basado en Simulated Annealing se muestra en el

algoritmo 1.

Para comenzar, se establece una temperatura inicial alta y una solución inicial aleatoria, lineas 1 a 3 del pseudocódigo 1. Luego se realizan modificaciones a la solución actual generando un conjunto de vecinos, como se muestra en la linea 27. Dentro de este vecindario, se elije un posible candidato de manera aleatoria. Como se aprecia en la linea 28 del algoritmo1, si dicho candidato es mejor que la solución actual, entonces se selecciona. Por el contrario, si el candidato empeora la solución actual, éste se debe someter a un testeo probabilístico en relación a la temperatura actual, linea 31. Este testeo decide si es seleccionado, linea 32, o se debe intentar elegir otro vecino. Con cada iteración la temperatura va bajando, linea 38, y una vez se haya iterado una cierta cantidad de veces la temperatura se reinicia a la inicial.

El algoritmo SA permite realizar movimientos que empeoran la calidad de las soluciones de manera de escapar de óptimos locales. Se debe entender como movimiento cualquier cambio que se le realice a la solución actual para moverse en el espacio de búsqueda. De esta forma, se permite pasar por malas soluciones para lograr encontrar soluciones óptimás y no estancarse en un área del espacio de búsqueda. Este proceso de estancamiento ocurre cuando el algoritmo se encuentra en bajas temperaturas y las opciones de movimiento no generan mejoras de calidad. La temperatura acá descrita se debe entender como una variable que permite la ejecución de movimientos que deterioren la calidad de las soluciones.

Para realizar esto se modela una temperatura T inicialmente alta, que permite con alta probabilidad la aceptación de movimientos que empeoran la calidad de las soluciones. Su valor es controlado con enfriamientos sucesivos en cada iteración al ser actualizada con una regla que pondera la temperatura actual por un parámetro $\alpha \in [0{,}8-0{,}99]$. A medida que la temperatura se reduce, también se reduce la probabilidad de aceptar movimientos que empeoren la calidad de la solución. De esta forma, se guía la solución a mejores espacios de búsqueda. Para no caer en un estancamiento, se realizan recalentamientos bruscos y periódicos actualizando la T a su valor inicial. La ejecución de un movimiento que genera un beneficio respecto a la calidad de la solución siempre se realiza. En caso de ser perjudicial, su aceptación se condiciona a una probabilidad dada por $P = e^{\Delta_{f.e.}/T}$. Donde $\Delta_{f.e.}$ corresponde a la diferencia entre la calidad de la solución inicial respecto de la calidad de la solución final.

3.2. Simulated Annealing para el problema de recolección de leche con mezclas y selección

El algoritmo construido es iterativo y está basado en Simulated Annealing. Posee una fase inicial de pre-procesamiento, una fase de construcción y finalmente una de post-procesamiento. A continuación se describen las componentes del algoritmo propuesto, detallando su estructura.

3.2.1. Representación

Dado que el objetivo del problema de recolección de leche con selección y mezclas es encontrar las rutas óptimás de cada camión, conocer qué granjas se visitan y en que orden, la representación utilizada corresponde a listas, ya que en ellas se puede apreciar tanto las granjas visitadas como su orden. A continuación se explica en detalle la representación usada.

Ruta de un camión

Se define como la ruta de un camión al conjunto de nodos visitados por éste, partiendo y terminando en la planta. Se utiliza un vector ordenado de valores enteros para representar la ruta de cada camión. Cada entero contenido en el vector corresponde al identificador de un nodo y su posición define el orden en que son visitadas por el camión. El identificador 0 corresponde a la planta y cada ruta siempre debe comenzar y terminar en ella. Por lo tanto un ejemplo de la ruta de un camión corresponde a:

$$\text{Ruta } k_2 = [0, 8, 3, 0]$$

donde el camión (k_2) comienza su recorrido en la planta, a continuación visita la granja 8, luego la 3 y vuelve a la planta.

3.2.2. Solución

La representación de la solución corresponde a un conjunto de las listas mencionadas en la sección anterior. Con esta representación es posible identificar qué granjas son visitadas y en qué orden. Por otro lado, generar una lista para cada camión obliga a generar soluciones que usen todos los camiones, permitiendo cumplir una de la restricciones del problema.

Considerando un conjunto de once nodos, con identificador de 1 a 10 para las granjas y 0 para la planta, una solución se puede representar de la siguiente forma:

$$\text{Ruta } k_1 = [0, 5, 1, 7, 0]$$
$$\text{Ruta } k_2 = [0, 8, 3, 0]$$

En ella el camión 1 (k_1) parte en la planta, visita la granja 5, luego la 1, continua con la 7 y finaliza en la planta. Por otro lado, el camión 2 (k_2) realiza el mismo recorrido descrito en la sección anterior. En este caso no se visitaron las granjas $2, 4, 6, 9, 10$. Además, es posible observar que los camiones no visitan necesariamente la misma cantidad de granjas, es por esto que el largo de las listas de los camiones no es el mismo. Basándose en este mismo ejemplo, la representación de una solución corresponde a:

$$\text{Rutas } = ([0, 5, 1, 7, 0], [0, 8, 3, 0])$$

3.2.3. Función de evaluación

La función de evaluación utilizada a lo largo del algoritmo corresponde a la que se aprecia en la ecuación 3.3. La función de evaluación utilizada considera la ganancia total, que corresponde a la diferencia entre la ganancia por la leche recolectada menos los costos de traslado asociados. La función de evaluación corresponde a la función objetivo del problema pues la propuesta trabaja sólo con soluciones factibles.

$$F.E. = \sum_{r \in T} \sum_{t \in T} L^r v^{tr} - C \sum_{k \in K} \sum_{(i,j) \in A} d_{ij} x_{ij}^k \tag{3.1}$$

3.2.4. Factibilidad

La factibilidad de las soluciones del problema de recolección de leche con selección y mezclas viene dada por la satisfacción de las cuotas de leche de cada tipo requeridas por la planta. Para satisfacer dichas restricciones es posible realizar mezclas de leche. En este caso particular se permite realizar mezclas de distintos tipos de calidades de leche tanto en los camiones como en la planta. A continuación se explican ambos casos.

Mezcla en un camión

La mezcla dentro de un camión se ve reflejada en la posibilidad de visitar granjas de distintas calidades. Esto hace que disminuya la calidad de la leche recogida y, por ende, su valor, debido a que se pierde la oportunidad de ser vendida como su tipo y pasa a ser vendida a un menor precio.

Continuando con el ejemplo presentado anteriormente, consideremos que las granjas productoras de leche tipo A son: $n^A = \{1, 2, 3, 4, 5, 6, 7\}$, las granjas que producen leche tipo B son $n^B = \{8, 9\}$ y la que produce leche tipo C es $n^C = \{10\}$

Por lo tanto, los camiones k_1 y k_2 recolectan leche tipo A y B respectivamente. El camión k_1 visita en su recorrido solo granjas de tipo A (nodos: 5, 1, 7). Por el contrario el camión k_2 visita granjas de tipo A y B (nodos: 3 y 8). Dado que $A > B$, es decir, A es una calidad mejor que B, la leche resultante dentro del camión k_2 corresponde a leche B. En esta solución la producción de leche del nodo n_3 sera multiplicada por el valor de leche B, el cual es menor al valor de venta de A y, por lo tanto, se pierde la posibilidad de venderla como tipo A y generar mayor ganancia.

Mezcla en la planta

En esta versión del problema también es posible realizar mezcla de leche en la planta procesadora. Este proceso se ejecuta una vez que se hayan realizado todos los recorridos y se tenga el total de leche de cada tipo recaudada por la flota de camiones. El proceso de mezcla en

planta se realiza siempre y cuando aún sea necesario satisfacer alguna demanda pendiente. La redistribución y mezcla de leche se realiza considerando, desde la leche de peor calidad a la mejor calidad, dado que la leche de mejor calidad se puede utilizar para suplir cualquier otro nivel de calidad, en cambio la leche de peor tipo solo puede suplir la demanda de si misma.

Considerando que las granjas del ejemplo anterior producen las cantidades de leche que se muestran en la tabla 3.1. Se tiene que el camión k_1 recolecta $35L$ de leche tipo A y el camión k_2 lleva a la planta $45L$ de leche de calidad B. Si se considera que las demandas de la planta corresponden a $P^A = 35L$, $P^B = 35L$ y $P^C = 5L$, entonces las leches recolectadas se utilizarían de la siguiente manera: $5L$ de leche B recolectada por el camión k_2 se utilizan para suplir la demanda de leche tipo C. Los $40L$ de leche tipo B restantes se utilizan para cumplir la demanda de tipo de leche B. Finalmente todo lo recolectado por el camión k_1 se utiliza como leche A pues es el mayor valor que se puede obtener de ésta.

Leche tipo A				Leche tipo B		Leche tipo C	
nodo	prod.	nodo	prod.	nodo	prod.	nodo	prod.
n_1	10	n_5	15	n_8	25	n_{10}	15
n_2	5	n_6	20	n_9	10		
n_3	20	n_7	10				
n_4	5						

Tabla 3.1: Producción de cada nodo según calidad de leche.

En esta solución se sobreabastece la planta con $5L$ de leche tipo B, pero dado que las demandas establecen únicamente un mínimo nivel a recolectar y se busca obtener el mayor beneficio, esto no genera un problema a menos que ir a buscar esa leche implique una perdida para la empresa.

3.2.5. Estructura

El algoritmo 2 muestra la estructura general de la propuesta de solución implementada.

Algoritmo 2: ALGORITMO SIMULATED ANNELING

Datos: μ número de reinicios
τ número de mejoras
β tamaño lista candidata
γ nodos a eliminar
ε multiplicador del numero de nodos que no se pudieron agregar

Resultado: *solucionMejor*: Mejor solución encontrada

```
1  PRE-PROCESAMIENTO: agrupar granjas según calidad - crear demandas ficticias
2  mientras no se alcance μ hacer
3      solucionActual = construcción(β, corte)
4      solucionActual = reordenarNodos()
5      mientras no se alcance τ hacer
6          solucionAnterior = solucionActual
7          solucionActual = romperDemandas(γ)
8          solucionActual = agregarNodos(ε)
9          si solucionActual es factible entonces
10             solucionActual = eliminarNodos()
11             solucionActual = reordenarNodos()
12         en otro caso
13             solucionActual = solucionAnterior
14         fin
15         si FE(solucionActual) < FE(solucionMejor) entonces
16             solucionMejor = solucionActual
17         fin
18         temperatura = temperatura * α
19     fin
20 fin
21 devolver solucionMejor
```

Se comienza con un proceso de pre-procesamiento en el que se listan las granjas de manera ordenada según el tipo de leche que producen. A continuación, mientras no se alcance el criterio de término, se ejecuta repetitivamente un proceso de construcción de solución seguido de una fase de post-procesamiento. La fase de post-procesamiento considera un conjunto *tau* de repeticiones en las cuales utiliza cuatro movimientos que buscan mejorar la solución inicial para acercarse lo más posible al óptimo global. Las fases de construcción y post-procesamiento se iteran μ y τ veces respectivamente. Iterar varias veces la construcción permite reiniciar la solución inicial y recomenzar el proceso en otro punto del espacio de búsqueda, por otro lado, iterar varias veces el post-procesamiento busca mejorar la solución inicial para acercarse lo más posible al máximo global.

A continuación se describen en detalle cada una de las fases del algoritmo propuesto.

3.2.6. Pre-procesamiento

En esta fase se insertan las granjas existentes en una lista ordenada según calidad de leche para trabajar con ellas con mayor facilidad.

3.2.7. Construcción

El algoritmo de construcción propuesto considera dos etapas principales.

En la primera etapa se utiliza un algoritmo pseudo-voraz (Greedy) con ruleta para seleccionar el siguiente nodo a insertar en la ruta actual.

La segunda etapa ocurre cuando no se logra suplir todas las demandas tras haber utilizado toda la flota de camiones y consiste en agregar nodos a las rutas que aún posean espacio, respetando el tipo de leche que éstas recolectan.

En el algoritmo 3 se presenta el pseudocódigo de la fase constructiva.

Algoritmo 3: CONSTRUCCION

Datos: β tamaño de lista candidata

corte si se acota o no la lista candidata del primero nodo.

Resultado: *solucion*: Solución inicial factible

```
1  CONSTRUCCION:
2  rutaActual = solucion.inicializarRuta(t_0)
3  mientras no se cumplan todas las demandas or no se usen todos los camiones hacer
4      nodoActual = rutaActual.back()
5      listaCandidata = generarListaCandidata(β, corte, nodoActual, distancia)
6      nodoSeleccionado = ruleta(listaCandidata, producción)
7      rutaActual = rutaActual→agregarNodo(nodoSeleccionado)
8      si camión lleno or demanda actual satisfecha entonces
9          rutaActual = solución→agregarRuta(t_insatisfecho))
10     fin
11     solución→actualizarSolucion()
12 fin
13 REPARACION:
14 mientras no se cumplan todas las demandas hacer
15     rutaActual = solucion→obtenerRutasConEspacio.back()
16     nodoActual = (rutaActual − nodoFinal).back()
17     listaCandidata = generarListaCandidata(β, nodoActual, distancia)
18     nodoSeleccionado = ruleta(listaCandidata, producción)
19     rutaActual = rutaActual→agregarNodo(nodoSeleccionado)
20     solucion→actualizarSolucion()
21 fin
22 devolver solucion
```

El algoritmo pseudo-voraz comienza con el armado de rutas que recolectan leche tipo *A* que es de mejor calidad. Una vez que se satisface la demanda de dicha calidad se continua con leche tipo *B* en un nuevo camión y así sucesivamente con cada calidad de leche existente. En caso de que un camión se llene, se continua construyendo la ruta del siguiente camión como

se aprecia en la linea 8 del algoritmo.

La selección del nodo a incluir en la ruta en cada paso se realiza de la siguiente manera. El algoritmo pseudo-voraz genera una lista de candidatos en cada iteración. En esta lista candidata se incluyen todos los nodos cuya producción no supera la capacidad disponible del tanque del camión y produzcan leche de igual o mejor calidad a la leche que se está recolectando. Si el nodo a insertar corresponde a la primera granja de la ruta, se utiliza la lista completa con el objetivo de poder comenzar en cualquier punto del espacio de búsqueda cuando el parámetro *corte* es falso. Por el contrario, si *corte* es verdadero, al igual que cuando ya existen granjas en la ruta, se consideran solo los β nodos más cercanos. De esta manera, se busca que el proceso de construcción genere rutas con bajos costos de transporte.

Una vez obtenida la lista candidata, se procede a seleccionar un nodo usando un procedimiento que favorezca a los nodos de mayor producción. Este procedimiento está basado en una ruleta que asigna a cada nodo una probabilidad de ser seleccionado proporcional a su producción. La ecuación 3.2 muestra el criterio de asignación de probabilidades de selección a los nodos.

$$\text{Probabilidad de elección del nodo i} = \frac{\text{Producción de leche del nodo i}}{\text{Producción de leche de los } \beta \text{ nodos}} \tag{3.2}$$

Según este esquema, los grandes productores de leche se escogen probablemente con mayor frecuencia, mientras que los pequeños productores, no se escogen más que de vez en cuando. De esta manera, se busca suplir más rápidamente las demandas de la planta, ya que al seleccionar nodos con mayor producción, será necesario visitar menos granjas para conseguir satisfacer las demandas y, por tanto, se tendrán menos costos de transporte.

En la figura 3.1 se ilustra una ruleta de selección para los $\beta = 5$ nodos más cercanos al nodo 7 (N_7). Utilizando la tabla 3.1, la producción total generada por los 5 nodos corresponde a $70L$. Tanto N_1, como N_2 poseen una probabilidad de ser elegidos menor al 15 % debido a sus producciones. Los nodos N_3 y N_6 obtienen una probabilidad de ser elegidos cercana al 30 % ya que producen mayor cantidad de leche. Por su parte, N_5 que produce $15L$ posee una probabilidad de selección igual a un 21 %, correspondiente a $\frac{15}{70}$.

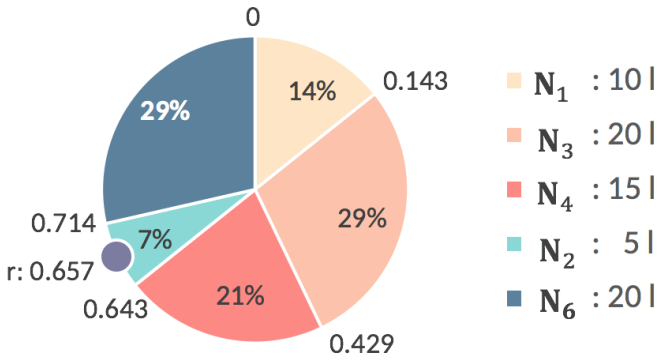


Figura 3.1: Ruleta sesgada para la lista candidata de n_7.

Una vez generada la lista candidata, se procede a utilizar un criterio de selección aleatorio. Esto requiere la generación de un número aleatorio r_1 donde $0 < r_1 \leq 1{,}0$. En el ejemplo de la figura 3.1, el valor de r_1 corresponde a $0,657$ por lo tanto la granja a visitar es la granja 2.

Una vez terminada esta etapa del proceso, en caso de que la solución confeccionada no logre suplir las demandas, se procede a reparar las rutas como se aprecia en las lineas 14 a 21 del algoritmo 3. Para esto se aplica el mismo procedimiento anteriormente descrito, es decir, se agregan granjas a las rutas que aun posean espacio, respetando el tipo de leche que estas recolectan. Antes de agregar un nodo se debe eliminar el viaje de vuelta a la planta como se ve en la linea 16 del pseudocódigo.

Con este procedimiento de construcción, sera posible obtener en cada nuevo reinicio una solución factible, de buena calidad y diferente a las anteriores.

En el caso que no se consideren niveles de demandas mínimos, se crean demandas ficticias utilizando una ruleta en base a la cantidad de producción en los nodos y el precio de cada tipo de leche. Al seleccionar un tipo se agrega a las demandas $PD * capacidadDelCamion$ leche de ese tipo. Esto se realiza reiteradamente, actualizando la cantidad de leche producida, hasta que ya no queden camiones. La creación de las demandas ficticias se debe a que los pasos de construcción se realizan mientras no se cumplan todas las demandas, como se aprecia en la linea 3 y 14 del algoritmo 3. Además, se utilizan las demandas como forma de verificar cuanto dejar de iterar en varias fases del algoritmo pospuesto, explicadas en las siguientes

secciones. Por esta razón si las demandas son cero, hay fases que no logran su objetivo entrando en ciclos infinitos.

El proceso de crear demandas ficticias se realiza cada vez que se va a crear una nueva solución inicial, con esto se evita crear una demanda ficticia general para la instancia y se logra explorar más el espacio de búsqueda.

3.2.8. Post-procesamiento

El objetivo principal de la fase de post-procesamiento es mejorar la calidad de las soluciones obtenidas en la fase de construcción. Esta fase está compuesta por cuatro movimientos, los cuales modifican las rutas de los camiones de distintas maneras. Estos movimientos se encargan de romper demandas, agregar nodos, reordenar nodos y eliminar nodos. En los dos primeros, es decir, romper demandas y agregar nodos, se utiliza el esquema Simulated Annealing y solo en caso de que el procedimiento agregar nodos logre volver a una solución factible, se continua con los últimos dos movimientos. De lo contrario, se vuelve a la solución anterior. Los movimientos utilizados, así como su descripción se presentan en esta sección.

Romper Demandas

El primer paso después de haber generado una solución inicial factible en la construcción, corresponde a romper sus restricciones satisfechas, es decir, volverla infactible. Para esto, se eliminan nodos de la solución hasta que el mínimo demandado se deje de cumplir y se hayan eliminado más de γ nodos. El objetivo principal de realizar este movimiento es generar espacio en las rutas que permita posteriormente agregar nuevos nodos hasta volver a satisfacer las demandas. Con esto se vuelve a un estado de factibilidad y se permite la diversificación, es decir, la exploración de diferentes regiones del espacio de búsqueda que pueden ser interesantes en la búsqueda de la solución.

Para realizar este movimiento, como se muestra en el Algoritmo 4, se comienza seleccionando de manera estocástica la ruta de un camión. Para esto, se recurre a un número aleatorio $r_2 \in [1, rutas.size()]$ siendo posible seleccionar cualquier ruta. Luego, si la ruta visita más de una granja se procede a seleccionar una de ellas por medio de otro valor aleatorio $r_3 \in [2, rutas.nodos() - 1]$. Esto permite seleccionar cualquier nodo que sea granja y no planta, es decir, sin considerar el primer y último nodo de cada ruta.

Algoritmo 4: ROMPER DEMANDAS

Datos: γ cantidad mínima de nodos a eliminar

Resultado: *solucion*: Solución infactible

```
1  mientras no se cumplan las demandas and se hayan eliminado γ nodos hacer
2      rutaAEliminar = solucion.rutas[r2]
3      si rutaAEliminar.size()>1 entonces
4          nodoAEliminar = rutaAEliminar.nodos[r3]
5          delta = calcularDelta()
6          si delta > 0 entonces
7              rutaAEliminar-> eliminarNodo(nodoAEliminar)
8          fin
9          en otro caso
10             si e^(delta/temperatura) > r4 entonces
11                 rutaAEliminar→eliminarNodo(nodoAEliminar)
12             fin
13         fin
14     fin
15     solucion→actualizarSolucion()
16 fin
17 devolver solucion
```

Una vez seleccionado el *nodoAEliminar*, como se muestra en la linea 5 del algoritmo 4, se calcula el deterioro $\Delta_{f.e.}$ asociado a la eliminación del nodo, es decir, cuanto cambia la solución al eliminar dicha granja. Para disminuir los costos de evaluación computacional, evitando calcular toda la función de evaluación, se procede a calcular solo el cambio, el

cual corresponde a una disminución en la producción total y un cambio en las distancias recorridas como se muestra en la ecuación 3.3.

$$\Delta_{f.e.} = -\{nodoActual.produccion + distancia(nodoAnterior, nodoSiguiente) - [distancia(nodoAnterior, nodoActual) + distancia(nodoActual, nodoSiguiente)]\} \tag{3.3}$$

donde el *nodoActual* corresponde al *nodoAEliminar*.

A continuación se desarrolla un ejemplo hasta obtener la solución final utilizada en los ejemplos anteriores. Considerando la ruta inicial construida para el camión k_1 correspondiente al recorrido $0, 7, 2, 6, 1, 0$ como se ilustra en la figura 3.2. Si se desea eliminar el *nodo*6, se disminuirá la producción total en $20L$. Además se reducirá la distancia total recorrida en el trayecto entre el nodo n_2 y el nodo n_6 y el comprendido entre el nodo n_6 y el nodo n_1. Por otro lado, la distancia total recorrida aumentará en el trayecto entre el nodo n_2 y el nodo n_1.

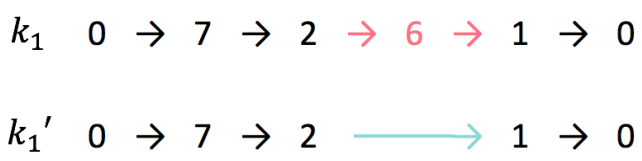


Figura 3.2: Cálculo de $\Delta_{f.e.}$ para la eliminación de n_6.

Si el $\Delta_{f.e.}$ es positivo, entonces el movimiento se acepta inmediatamente, de lo contrario, como se presenta en la linea 10 del pseudocódigo del algoritmo 4, se debe calcular la probabilidad de aceptación $P = e^{\Delta/T}$. Si P es mayor que un valor aleatorio $r_4 \in [0, 1]$ se acepta el movimiento, de lo contrario se debe volver a realizar el movimiento. Esta es una de las partes del algoritmo que guardan relación con el algoritmo Simulated Annealing.

Suponiendo que $\Delta_{f.e.}$ es positivo para el ejemplo, el resultado del movimiento es la ruta k_1' equivalente a $0, 7, 2, 1, 0$. En esta nueva ruta, se posee un total de $25L$ recolectados, por lo que deja de cumplir la demanda P^A de la planta correspondiente a $35L$. Si $\gamma = 1$, entonces se puede continuar con el siguiente movimiento.

Agregar Nodos

El objetivo de este movimiento es volver a cumplir con las demandas mínimas y así obtener una solución factible. De esta manera, se permite la exploración de nuevas regiones en el espacio de búsqueda.

La selección e inserción de un nodo, al igual que en el movimiento anterior, se realiza por medio de un esquema basado en Simulated Annealing. Primero se selecciona la ruta que transporte la mejor calidad de leche y que posea espacio para transportar la producción de alguna granja de la lista candidata disponible para esa ruta. La lista candidata corresponde a todas las granjas que no hayan sido visitadas y cuya producción sea de igual o mejor calidad que la recolectada por la ruta seleccionada. Además, para que una granja pertenezca a la lista candidata disponible, su producción debe ser menor o igual a la capacidad disponible que posea el camión de la ruta seleccionada. La lista candidata se vuelve a calcular cada vez que se inserta un nodo en la ruta ya que la capacidad disponible se ve afectada.

A continuación, se debe seleccionar un punto en la *rutaSeleccionada*. Para esto se recurre a un valor aleatorio $r_5 \in [1, ruta.size() - 1]$ correspondiente al *indiceSeleccionado*. Para el *indiceSeleccionado* se busca el mejor candidato dentro de la lista candidata, que corresponde a la granja que genere un mayor beneficio. Este beneficio se calcula para todo nodo perteneciente a la lista candidata y se obtiene a partir de la expresión 3.4.

$$\begin{aligned} beneficioo\Delta_{f.e.} = \{&nodoActual.produccion + distancia(nodoAnterior, nodoSiguiente) \\ &-[distancia(nodoAnterior, nodoActual) + distancia(nodoActual, nodoSiguiente)]\} \end{aligned} \tag{3.4}$$

donde el *nodoActual* corresponde a una granja de la lista candidata. El *nodoSiguiente* y *nodoAnterior* corresponden a nodos adyacentes en la *rutaSeleccionada*. El *nodoAInsertar* es el que obtenga el beneficio más alto.

Como es posible apreciar, este beneficio es similar al $\Delta_{f.e.}$ de la sección 3.2.8 y mide cuánto cambia la función de evaluación al insertar el *nodoAInsertar*. Si este valor es positivo, su

inserción es aceptada inmediatamente, de lo contrario se procede a calcular la probabilidad $P = e^{\Delta_{f.e.}/T}$ y compararlo con un valor aleatorio $r_6 \in [0, 1]$. Si P es mayor que r_6 la *nodoAInsertar* se inserta en el *indiceS eleccionado*, de lo contrario se vuelve al momento en que se selecciona una ruta con espacio disponible.

El pseudocódigo para este movimiento se muestra en el algoritmo 5.

Algoritmo 5: AGREGAR NODOS

Datos: ε multiplicador del numero de nodos que no se pudieron agregar

Resultado: *solucion*: Solución factible con exceso de nodos

```
1  mientras existan rutas con espacio hacer
2      si rutaS eleccionada.size() == 2 entonces
3          rutaS eleccionada.tipo = nuevoTipoRuta()
4      fin
5      indiceS eleccionado = rutaS eleccionada[r5]
6      nodoAAgregar = obtenerMejorOpcion(indiceS eleccionado, beneficio)
7      delta = calcularDelta()
8      si delta > 0 entonces
9          rutaS eleccionada→agregarNodo(nodoAAgregar, indiceS eleccionado)
10     fin
11     en otro caso
12         si e^(delta/temperatura) > r6 entonces
13             rutaS eleccionada→agregarNodo(nodoAAgregar, indiceS eleccionado)
14         fin
15     fin
16     solucion→actualizarSolucion()
17     si no se agregan ε * rutaS eleccionada.size() nodos entonces
18         temperatura = temperaturaInicial
19     fin
20 fin
21 devolver solucion
```

En ciertas circunstancias es posible que se llegue a un estancamiento al intentar insertar

repetidamente sin éxito algún nodo a cierta ruta. Para evitar entrar en un ciclo infinito se recurre a recalentar el sistema. Por lo tanto, si se realizan $\varepsilon * ruta.size()$ inserciones sin éxito, la temperatura del sistema vuelve a su valor inicial, como se aprecia en la linea 17 del algoritmo 5. Esto permite aceptar con mayor facilidad los movimientos que empeoren la solución final, además de permitir la diversificación y darle una holgura al algoritmo antes de recalentar.

Continuando con el ejemplo, considerando que la capacidad máxima de ambos camiones de la flota corresponde a $45L$, entonces solo es posible que el camión k_1 visite más granjas, ya que posee una capacidad disponible de $20L$. En cambio el camión k_2 se encuentra lleno. La lista candidata para esta situación corresponde a los nodos $4, 5, 6$. El resto de los nodos ya se visita en la otra ruta (nodos: $1, 2, 3, 7, 8$) o produce leche de una calidad inferior a la del camión k_1 el cual recolecta leche A (nodos: 9, 10). También son descartados los nodos que producen más leche que el espacio disponible en el camión. Si se intenta agregar n_5 entonces la función de evaluación se puede calcular solo considerando lo descrito en la ecuación 3.4 tal como se ilustra en la figura 3.3.

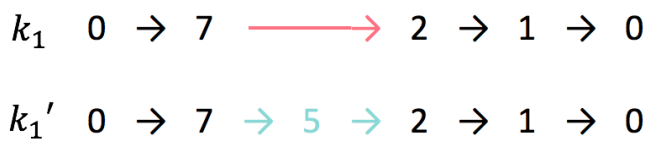


Figura 3.3: Cálculo de $\Delta_{f.e.}$ al agregar n_5.

Considerando que se acepta agregar el nodo n_5, bajo las reglas de un esquema Simulated Annealing, la solución actual del ejemplo corresponde a: $([0, 7, 5, 2, 1, 0], [0, 8, 3, 0])$ con lo cual se esta recolectando $40L$ de leche tipo A y $45L$ de leche tipo B. Al encontrarse en una situación de factibilidad, es posible continuar con los siguientes movimientos, de lo contrario se debe volver a la solución anterior y realizar nuevamente los movimientos.

Como es posible apreciar en la linea 2 del algoritmo 5, en el movimiento Agregar Nodos, se realiza un cambio en el tipo de leche que se recolecta si la ruta no posee ninguna granja en su recorrido. Esto se realiza tomando en cuenta todos los nodos que no han sido visitados y en base a su producción, multiplicada por el precio de venta de ese tipo de leche, se genera

una ruleta. A partir de ella, se selecciona un tipo y se utiliza como el nuevo tipo de leche que recolecta el camión. Todo esto solo si ya se esta cumpliendo la cuota mínima demandada de cada tipo. Esta forma de seleccionar el nuevo tipo de leche es similar a la forma en que se generan demandas ficticias como se explico al final de la sección 3.2.7

En este punto, como se aprecia en la linea 8 del pseudocódigo del algoritmo 2, solo si la solución obtenida tras el movimiento Agregar Nodos es factible, cumpliendo las demandas mínimas de la planta, se continua con el resto de los movimientos. De lo contrario se vuelve al punto antes de Romper Demandas y se realizan nuevamente los movimientos.

Reordenar Nodos

Este movimiento está enfocado en la intensificación y busca encontrar el mejor orden para los nodos dentro de cada ruta. Por lo tanto, es necesario considerar solo la distancia recorrida, pues la cantidad de leche recolectada no se ve afectada. Para esto se revisan todas las rutas, y por ruta se toma cada granja evaluando en qué posición de ese recorrido es más corto visitarla.

Algoritmo 6: REORDENAR NODOS

Resultado: *solucion*: Solución factible ordenada

```
1 para ruta ∈ rutas hacer
2     para nodoAMover ∈ ruta hacer
3         ruta− > eliminarNodo(nodoAMover)
4         indiceSeleccionado = ruta.probarInsercion(nodoAMover)
5         ruta− > agregarNodo(nodoAMover, indiceSeleccionado)
6     fin
7     solucion.actualizarSolucion()
8 fin
9 devolver solucion;
```

Como se aprecia en el pseudocódigo del algoritmo 6, se selecciona un *nodoAMover* de la ruta y se saca del recorrido. A continuación se calcula el aumento de distancia al insertar el

nodo en cada posición de la ruta como se muestra en la ecuación 3.5.

$$distanciaInsercion = distancia(nodoAnterior, nodoAMover) + distancia(nodoAMover, nodoSiguiente) \tag{3.5}$$

Continuando con el ejemplo, si para el camión k_1 se desea encontrar la mejor posición para n_7 se procede a eliminarlo de la ruta y calcular la *distanciaInsercion* de n_7 entre cada par de nodos como se presenta en la figura 3.4.

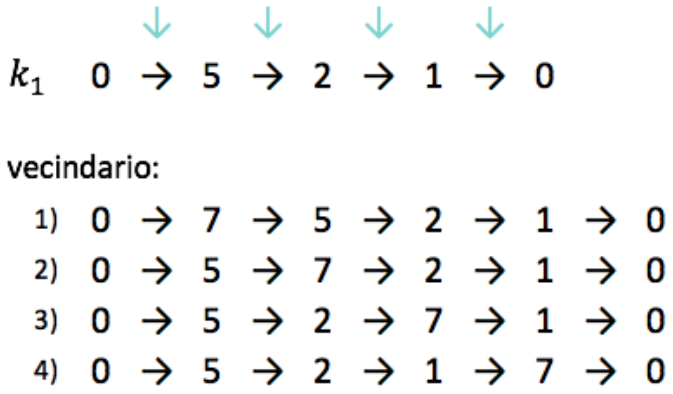


Figura 3.4: Candidatos para reordenar n_7 en la ruta del camión k_1.

En este movimiento no es necesario contar con una ruleta, ya que se selecciona el mejor vecino, es decir, el que obtenga una *distanciaInsercion* menor dado que se está buscando la ruta óptima. En el ejemplo de la figura 3.4, suponemos que la ruta más corta corresponde a la opción 4).

Remover Nodos

El último movimiento consiste en remover los nodos que no significan un beneficio para la solución, es decir, aquellos cuya ganancia por leche producida es menor que el costo de la distancia recorrida para visitarlo. Estos nodos se pueden eliminar, siempre y cuando no se dejen de satisfacer las demandas, ya que éste es el último movimiento y se espera obtener una solución final que cumpla todas las restricciones. Por lo tanto, este movimiento busca mejorar la solución, quitando todos aquellos nodos que sobren.

Algoritmo 7: ELIMINAR NODOS

Resultado: *solucion*: Solución factible con la cantidad justa de nodos

```
1  para ruta ∈ rutas hacer
2      mientras no se dejen de satisfacer las demandas hacer
3          listaCandidata = generarlistaCandidata(ruta);
4          si !listaCandidata.empty() entonces
5              nodoSeleccionado = ruleta(listaCandidata, β);
6              rutaActual = rutaActual− > eliminarNodo(nodoSeleccionado);
7          fin
8          solucion− > actualizarSolucion();
9      fin
10 fin
11 devolver solucion;
```

En el algoritmo 7 es posible apreciar que este movimiento también recurre a una ruleta sesgada. La lista de candidatos considera a todos los nodos cuya eliminación signifique un beneficio, es decir, que aumente la función de evaluación al quitarlo. Esto se evalúa igual que en el movimiento Romper Demandas utilizando la ecuación 3.3. Además, al eliminarlo se debe seguir cumpliendo las demandas mínimás de la planta. Esto se realiza evaluando todas las rutas, para así obtener una solución totalmente limpia de nodos innecesarios que pudieron ser agregados en el movimiento de Agregar Nodos.

Para finalizar con el ejemplo, se puede considerar que el nodo n_2, que produce solo $5L$ se encuentra apartado del recorrido del camión k_1 y por lo tanto no es beneficiosa su visita. Con la configuración actual de solución, correspondiente a: $([0, 5, 2, 1, 7, 0], [0, 8, 3, 0])$, se está entregando en la planta $40L$ de leche A y $45L$ de leche tipo B. En la planta se utilizan $5L$ de leche B como leche C para satisfacer la demanda de esa calidad. Los $40L$ restantes de leche B se utilizan para satisfacer los $35L$ de demanda tipo B, sobre abasteciendo la planta con $5L$ de leche B. Finalmente, ya que la demanda de leche de calidad A corresponde a $35L$, la planta es sobre abastecida con $5L$ de leche A y, por tanto, es posible eliminar n_2, el cual, como se mencionó anteriormente, se encuentra apartado del recorrido. La solución final corresponde a: $([0, 5, 1, 7, 0], [0, 8, 3, 0])$.

3.3. Resumen

El algoritmo implementado para este trabajo considera tres fases: pre-procesamiento, construcción, post-procesamiento. La primera fase es bastante sencilla, se encarga de realizar una lista de las granjas según el tipo de leche que producen. Luego, la construcción se encarga de generar una solución factible inicial de buena calidad considerando la producción y posición de cada nodo. Finalmente, la fase de post-procesamiento se divide en 4 movimientos: romper demandas, agregar nodos, reordenar y remover nodos. Los dos primeros movimientos utilizan Simulated Annealing tomando en cuenta la temperatura del sistema y el beneficio del movimiento al realizarlo. Por otro lado, los últimos dos movimientos se encargan de mejorar la solución al eliminar los nodos que no generan beneficio y ordenar las rutas de mejor manera. Todo esto se realiza reiteradas veces con el fin de explorar el espacio de búsqueda para acercarse lo más posible al máximo global.

Capítulo 4

Implementación

En este capítulo se presenta el análisis experimental realizado y los resultados obtenidos durante el proceso de evaluación del algoritmo propuesto. Se especifican los objetivos del proceso de experimentación, se explican las instancias de pruebas utilizadas, las características generales del proceso de experimentación y el entorno computacional donde se ejecutaron los experimentos.

A continuación, se exponen los resultados, se analizan diferentes componentes de la propuesta, se comparan los resultados propios con los resultados obtenidos utilizando un modelo de programación lineal entera y se evalúa el uso del algoritmo para resolver instancias de mayor tamaño.

4.1. Experimentos

En esta sección se especifican los objetivos del proceso de experimentación, las instancias de pruebas utilizadas y las características del proceso de experimentación realizado.

4.1.1. Objetivos

Los objetivos del proceso de experimentación de este trabajo de memoria se pueden resumir en los siguientes puntos:

- Estudiar la relevancia de los componentes más importantes del algoritmo propuesto. En este caso se han seleccionado los componentes asociados a la flexibilidad de los procesos de construcción y los movimientos propuestos que se encuentran basados en un esquema Simulated Annealing.
- Comparar los resultados obtenidos con los resultados obtenidos utilizando el solver CPLEX para resolver el modelo de programación lineal asociado al problema para un conjunto de instancias pequeñas.
- Evaluar la calidad de los resultados en instancias de mayor tamaño.

4.1.2. Instancias de Prueba

Para evaluar el algoritmo propuesto, se consideraron dos conjuntos de instancias de pruebas. El primer conjunto, que denominaremos instancias pequeñas, corresponden a un subconjunto de 40 nodos pertenecientes a la instancia utilizada por Paredes-Belmar et. al. en [29] 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 se poseen 3 calidades de leche, cuyos valores de venta corresponden a $L^A = 0{,}03u.m.$, $L^B = 0{,}021u.m.$, $L^C = 0{,}009u.m.$. Dados estos valores se considera que la leche de calidad A es mejor que la de tipo B y tipo C, siendo esta última la de peor calidad.

A continuación se presentan ambos conjuntos y se analizan sus características generales.

Como apoyo para el análisis, además de tablas, se utilizan boxplots.

Un boxplot o gráfico de caja es una forma estandarizada de mostrar la distribución de un conjunto de datos. A partir de este tipo de gráficos se puede obtener los valores extremos, cuartiles y los valores estadísticos erráticos o atípicos, más conocidos como outliers. Además, en los boxplots se puede apreciar fácilmente la dispersión los datos y si estos poseen una distribución simétrica o se encuentran sesgados. La figura 4.1 ilustra los componentes principales de un boxplot. El Rango Intercuartílico o RIC, corresponde al 50 % de los datos y se ilustra en forma de caja, en su interior se encuentra la mediana ilustrada con una linea naranja.

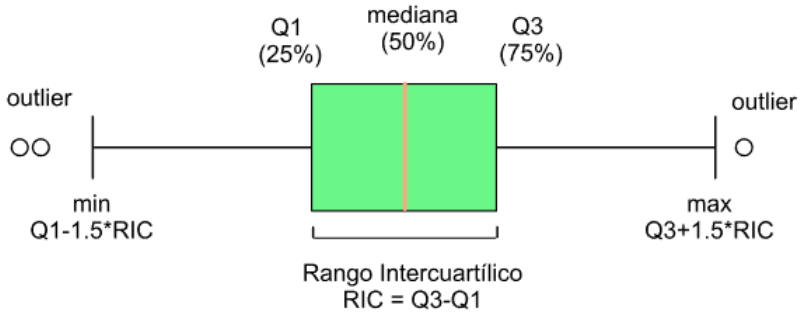


Figura 4.1: Componentes principales de un boxplot.
Fuente: Elaboración propia.

Instancias pequeñas

El conjunto de instancias pequeñas está compuesto por seis instancias. Todas estas instancias poseen 40 nodos. El conjunto de 40 granjas produce en total 197670L de leche, dividido en 61233L de leche tipo A, 56911L de leche tipo B y 79526L de leche tipo C. Con estas producciones se tiene que un 31 % de la leche es de calidad A, un 29 % es de tipo B y 40 % es de calidad C. Los detalles de producción y calidad de leche producida se detallan en la tabla 4.10 del anexo 4.2.4. En la figura 4.2 se presenta un boxplot de las producciones de los nodos separados por tipo de leche. En esta visualización es posible apreciar que la distribución de los nodos que producen leche tipo A es bastante uniforme. Por su parte, los nodos que producen leche de calidad B poseen una distribución similar a la de leche tipo A, pero se encuentran sesgados a una mayor producción. Los nodos que producen leche tipo

C también poseen un sesgo positivo, pero una distribución mayor a las otras dos calidades. Finalmente, es importante notar que existe un valor atípico por cada calidad de leche.

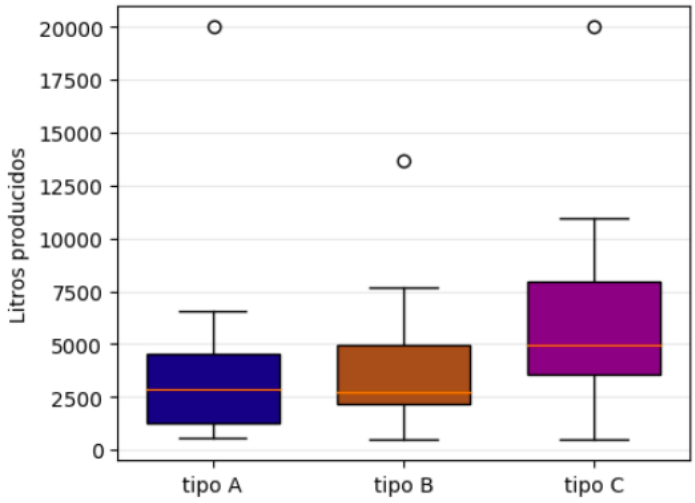


Figura 4.2: Producciones de nodos según la calidad de leche para las instancias pequeñas.

Los nodos considerados atípicos son analizados en la tabla 4.1 donde se presenta la producción y calidad de leche de cada nodo, las distancias promedios, menor, mayor y la distancia a la planta. La matriz de distancias para los 40 nodos se presenta en la tabla 4.11 del anexo 4.2.4. Los nodos 40 y 39 que producen $20000L$ cada uno se encuentran relativamente aislados, dado que los nodos más cercanos a ellos están a $143u.d.$ y $127u.d.$ respectivamente y si se analiza el resto de los nodos de la matriz, todos poseen al menos un vecino a menos de $27u.d.$. Por otro lado, el nodo 20 se encuentra relativamente integrado al resto de los nodos, estando en promedio a $108u.d.$ del resto.

Nodo	**Leche**	**Producción**	**Distancia**			
			Menor	Mayor	Promedio	a Planta
nodo 40	A	20000	143	360	305.2	305
nodo 20	B	13699	13	321	108.2	74
nodo 39	C	20000	127	303	249.6	248

Tabla 4.1: Características de los nodos con producción atípica.

En cada una de las instancias de prueba se trabajó con una flota de camiones distinta y las

demandas de leche por parte de la planta también son diferentes. En la tabla 4.2 se presenta dicha información en detalle.

Por cada una de las instancias numeradas en la primera columna de dicha tabla se presentan los miles de litros de leche de cada tipo demandados por la planta. Estos valores se encuentran agrupados bajo la columna *Demandas*. Cada columna P^x corresponde a los miles de litros de leche demandados de cada calidad y la columna total corresponde a la suma de todos los litros demandados. Estos últimos también se encuentran expresados en miles. A continuación, bajo la columna *Demanda/Total por tipo*, se presenta que porcentaje del total de leche producida de cada tipo es demandado por la planta. Por ejemplo, para la instancia 1, la planta demanda 60 mil litros de leche de tipo A, y estos corresponden a un 98 % del total de leche tipo A producida, equivalente a $60000/61233$. Por otro lado, las últimas dos columnas indican la cantidad de camiones que se posee en cada instancia y su capacidad expresada en miles de litros, la cual es igual para toda la flota.

Instancia	**Demandas**				**Demanda/Total por tipo**			**# cam.**	**Capac.**
	P^A	P^B	P^C	Total	P^A	P^B	P^C		
instancia 1	60	40	10	110	98 %	70 %	13 %	7	20
instancia 2	60	40	10	110	98 %	70 %	13 %	5	30
instancia 3	0	0	0	0	0 %	0 %	0 %	4	30
instancia 4	0	0	0	0	0 %	0 %	0 %	5	30
instancia 5	10	70	10	90	16 %	123 %	13 %	5	30
instancia 6	40	40	10	90	65 %	70 %	13 %	5	20

Tabla 4.2: Características de las instancias pequeñas.

Analizando las instancias más a fondo, podemos ver que en las dos primeras la planta demanda la misma cantidad de cada tipo de leche. La diferencia en este caso es la cantidad de camiones que se poseen y su capacidad. Con la flota de 7 camiones de $20000L$ que posee la primera instancia, se podría cargar un máximo de $140000L$. Por otro lado, con la flota de la segunda instancia se puede trasladar $10000L$ más. Es probable que las rutas de la segunda instancia involucren más nodos, pues aumenta la capacidad de los camiones. Por su parte, en la instancia 1 se debe construir las rutas de un mayor número de camiones.

Las instancias 3 y 4 no poseen demandas y precisamente esa es su dificultad. Al tener demandas nulas la restricción de cumplir con las demandas se pierde y, por tanto, el problema se relaja, pues cualquier ruta inicial es factible bajo el punto de vista de las demandas. Para el algoritmo implementado, esto genera una dificultad ya que en varias de las decisiones que toma el algoritmo vienen dadas por la información de demanda mínima.

La instancia 5, posee una demanda de leche de calidad B que es un 23 % mayor que la producción total de ese tipo de leche. Esto implica que, para suplir la demanda de leche tipo B, se tiene que utilizar leche de tipo A que es de mejor calidad. La mezcla de leche A y B se puede generar tanto dentro del camión como en la planta. Debido a que la capacidad de los camiones es tres veces mayor que la demanda de leche tipo A, probablemente se recurra a degradar a B cierta cantidad de leche A para aprovechar más el estanque del camión. Esta instancia también es la que posee mayor holgura en cuanto a capacidad máxima de transporte en comparación a la cantidad demandada, donde se tiene un 40 % de la capacidad de transporte total disponible para sobre abastecer a la planta. Considerando que la flota de camiones está compuesta por 5 vehículos con tanques de $30000L$, en una situación ideal, dos de ellos podrían quedar a disposición para sobre abastecer la planta.

Por la configuración de las demandas y las características de la flota de camiones de la sexta instancia, es poco probable que un camión transporte leche de tipo C. Para que un camión de la flota traslade leche tipo c, se tendría que ocupar al máximo la capacidad de cuatro camiones transportando leche de tipo A o B con lo cual se lograría suplir justo dichas demandas. Por lo tanto, dada la baja probabilidad de ocurrencia de la situación anterior, esta instancia invita a realizar mezcla en la planta para suplir la demanda de leche tipo C. En total se producen 118144 litros de leche tipo A y B, por lo tanto, los 90 litros demandados en esta instancia son aproximadamente un 76 % del total producido. Además, dado que se cuenta con una flota de 5 camiones con una capacidad de $20000L$ cada uno, se tiene una capacidad de transporte máxima de $100000L$, lo cual deja un 10 % de la capacidad total de trasporte disponible para sobre abastecer a la planta.

Instancias reales

Las instancias grandes están basadas en la distribución geográfica y producción de 500 granjas ubicadas en el sur de Chile. A partir de esta información se construyen 6 conjuntos. Un conjunto corresponde al total de nodos y los otros cinco conjuntos se construyen aplicando un algoritmo de agrupamiento que divide las 500 granjas en 5 grupos. Para el proceso de agrupamiento se utilizó el algoritmo k-means.

El algoritmo k-means fue publicado por primera vez por en 1982 [25] y es un método de agrupamiento simple y popular. El objetivo de k-means es agrupar puntos de datos similares y encontrar patrones entre ellos. Para lograr esto, k-means busca una cantidad definida k de grupos en el conjunto de puntos disponibles. Por lo tanto, k es el número de centros que se necesitan encontrar, donde se considera que un punto pertenece al grupo que tenga su centro más cercano a él. Con esto se busca mantener los grupos lo más concentrados posible obteniendo una distancia promedio entre nodos de cada conjunto igual a $57u.d$. Los conjuntos $c1$, $c4$, $c5$ se encuentran a una distancia promedio cercana a los $50u.d$. de la planta, en cambio, los conjuntos $c2$ y $c3$ se encuentran más alejados, estando en promedio a más de $85u.d$..

Conjunto	**# Nodos**	**Tipo A**	**Tipo B**	**Tipo C**
c1	128	**204262**	157972	121935
c2	143	135762	**161120**	114726
c3	142	138939	**156584**	137857
c4	31	50757	33869	**66416**
c5	56	81059	**115171**	75443
a500	500	516308	**669502**	587364

Tabla 4.3: Producciones totales de cada calidad de leche por conjunto.

En la tabla 4.3 se presentan los litros totales de leche producida de cada tipo por conjunto y el número total de nodos que poseen. En negrita se marca el tipo de leche que posee mayor producción en cada conjunto, siendo la de calidad B la que suele ser mayor. Solamente en el primer conjunto la mayor producción de leche se encuentra en el tipo A y en el conjunto 4 se

encuentra en el tipo C.

La mayor diferencia de producciones totales se encuentra en el primer conjunto, donde se tienen cerca de $50000L$ más de leche tipo A que leche B y $70000L$ más que de leche C. El resto de los conjuntos posee una diferencia de producción entre cada tipo que no supera los $300000L$. Por su parte, el conjunto mayor, de 500 nodos, produce cerca de $150000L$ más de leche tipo B que leche tipo A y $80000L$ más que de leche tipo C.

Es importante mencionar que el tipo de leche producido por cada nodo en el conjunto de 500 granjas es diferente que el producido por esos nodos en los conjuntos generados por k-means. Esto se debe a que la asignación del tipo de leche fue realizada después de haber generado los conjuntos. Esta asignación se realizó de manera aleatoria solo considerando tener la misma cantidad de nodos por tipo de leche producida en cada conjunto. La creación de estos grupos se realizo para lograr generar un mayor número de instancias diferentes para probar el algoritmo implementado. No se pueden realizar comparaciones directas entre el conjunto de nodos generados por k-means y el conjunto de 500 nodos, pues los nodos difieren en el tipo de leche producida, lo cual genera configuraciones de rutas diferentes.

Para analizar las producciones de leche por tipo con mayor profundidad se generaron los gráficos de la figura 4.3. Los conjuntos $c2$ y $c3$ son los que poseen la mediana más baja, esto quiere decir que un 50 % de los nodos producen menos de $2000L$ de leche. En todos los conjuntos se tiene un sesgo positivo y outliers bastantes alejados del máximo. La excepción es el boxplot de la calidad B del conjunto $c4$ donde se tiene un sesgo negativo y, sin contar el outlier, todos los nodos producen menos de $5000L$. Este conjunto posee solo 31 siendo el más pequeño de todos y por ende el que menos producciones totales tiene, como se aprecia en la tabla 4.3.

Por otro lado, el conjunto $c5$ es el que posee la distribución más uniforme entre cada calidad de leche. Las producciones de leche tipo C poseen un sesgo positivo y la calidad B es la única que presenta outliers.

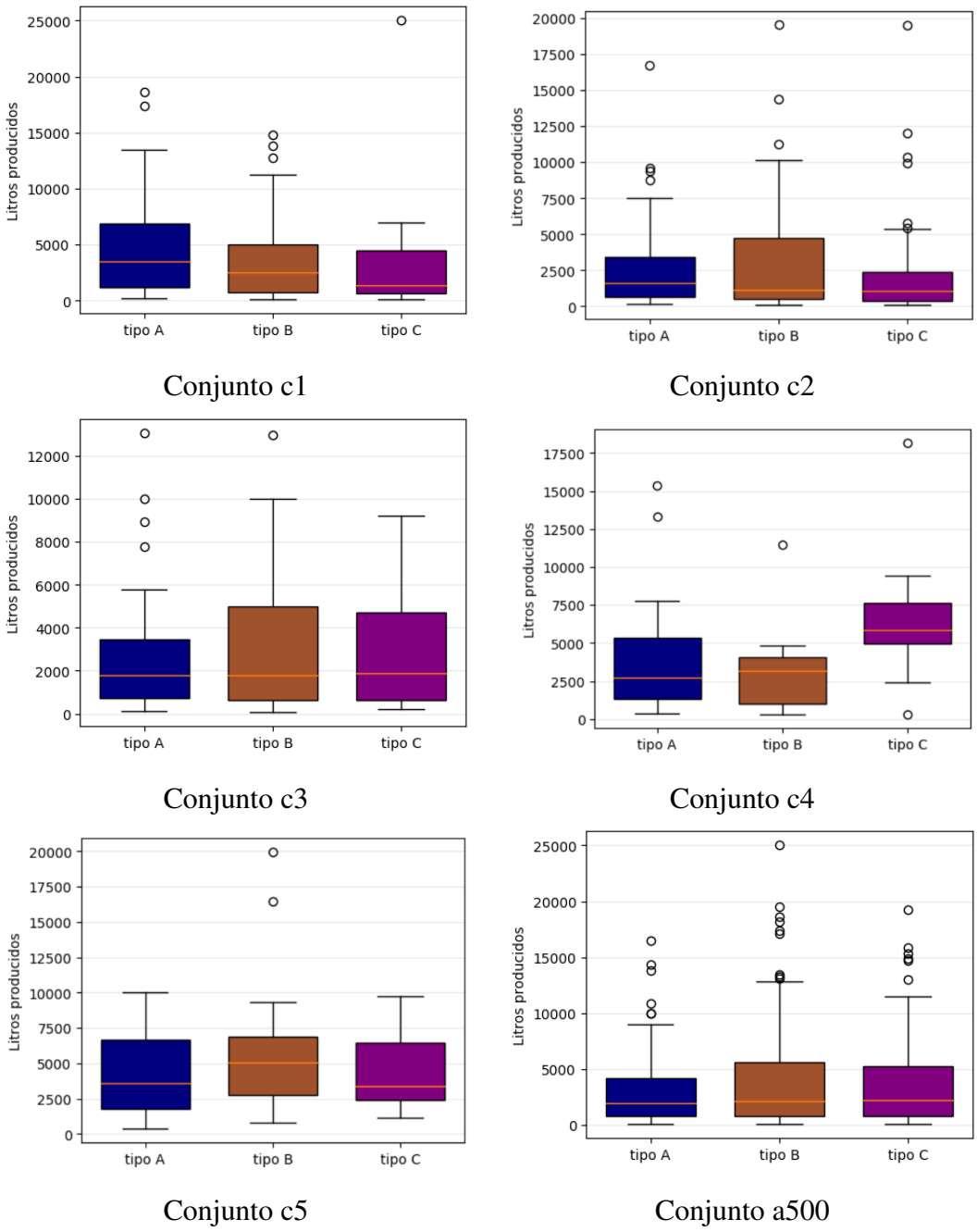


Figura 4.3: Litros producidos por cada nodo según el tipo de leche para cada conjunto.

Mirando los outliers del conjunto mayor, podemos ver que el nodo que produce cerca de 25000L es de calidad B en cambio dentro de los conjuntos armados por k-means este nodos se encuentra en el conjunto $c1$ y es de calidad C. A pesar de la gran cantidad de nodos y la

forma aleatoria de asignar la calidad de leche producida a cada nodo, en la instancia de 500 nodos las medianas se ubican bastante cerca, esto quiere decir que el 50 % de los nodos en cada calidad producen menos de 2000 litros de leche.

A continuación se explica como se generaron las instancias y cuáles son sus características principales.

La creación de instancias se realizó utilizando los 6 conjuntos que incluyen los 5 grupos generados por el proceso de agrupamiento y el conjunto de los 500 nodos totales. Para cada uno de los grupos de granjas se construyeron dos instancias de prueba. La primera de ellas considera una opción que requiere 16 % del total de leche tipo A, 123 % del total de leche tipo B y 13 % del total de leche tipo C producido por el conjunto de granjas correspondiente a cada uno de los grupos. La segunda instancia no especifica cuotas de leche para ninguno de los tipos producidos. En todas las instancias se considera solo el uso de camiones de $30000L$. La cantidad de camiones se asigna para cada problema considerando que la capacidad total de recolección debiera ser al menos un 10 % superior respecto a la producción total requerida. Se crearon esas instancias en base a las configuraciones de demandas de las instancias 4 y 5 del grupo de instancias pequeñas.

La tabla 4.4 presenta la misma información que la tabla 4.2 y una columna extra llamada *Nodos* la cual expresa la cantidad de nodos totales de cada instancia generado a partir del algoritmo k-means. Las últimas dos filas corresponden a una instancia con todos los nodos existentes.

Es importante notar que las instancias $c2$ y $c3$ están compuestas por casi la misma cantidad de nodos y sus demandas totales son equivalentes, pero las demandas por tipo varían levemente. Por otro lado, la instancia $c1$ posee cerca de 15 nodos menos que las otras dos nombradas, pero su demanda total es mayor en $10L$.

La instancia $c4$ posee una holgura del 34 %, pues su capacidad de transporte máxima corresponde a $90L$ y la demanda es de $60L$, convirtiéndola en la instancia grande con mayor holgura. El resto de las instancias poseen una holgura cercana al 10 %.

Instancia	# Nodos	Demandas				Demanda/Total por tipo			# Cam.	Capac.
		P^A	P^B	P^C	Total	P^A	P^B	P^C		
c1.0	128	0	0	0	0	0 %	0 %	0 %	9	30
c1.1	128	35	195	15	245	16 %	123 %	13 %	9	30
c2.0	143	0	0	0	0	0 %	0 %	0 %	9	30
c2.1	143	20	200	15	235	16 %	123 %	13 %	9	30
c3.0	142	0	0	0	0	0 %	0 %	0 %	9	30
c3.1	142	25	190	20	235	16 %	123 %	13 %	9	30
c4.0	31	0	0	0	0	0 %	0 %	0 %	3	30
c4.1	31	10	40	10	60	16 %	123 %	13 %	3	30
c5.0	56	0	0	0	0	0 %	0 %	0 %	6	30
c5.1	56	15	140	10	165	16 %	123 %	13 %	6	30
a500.0	500	0	0	0	0	0 %	0 %	0 %	35	30
a500.1	500	80	800	75	955	16 %	123 %	13 %	35	30

Tabla 4.4: Características de las instancias reales.

4.1.3. Sintonización

El proceso de sintonización tiene como propósito encontrar los mejores valores posibles para los parámetros y la posterior ejecución del algoritmo. El método de sintonización utilizado para esta memoria es ParamILS, el cual se basa en el proceso de sintonización manual. Propuesto en [18], ParamILS es un método de sintonización automático que utiliza búsqueda local iterativa para encontrar la mejor configuración de parámetros para un algoritmo metaheurístico utilizando como conjunto de entrenamiento un conjunto de problemas de prueba.

Para aplicar ParamILS en el algoritmo propuesto, se consideró como conjunto de entrenamiento las seis instancias de pruebas que componen el conjunto de instancias pequeñas previamente explicado en la sección 4.1.2. Además, es necesario definir los dominios deseados de cada uno de los parámetros, un conjunto finito de valores posibles para cada parámetro y un valor inicial para el proceso de sintonización. Dichos valores se presentan en la tabla 4.5. A continuación se explican los parámetros listados.

Parámetro	Valores posibles	Valor inicial
β	$\{1, 2, 3, 4, 5, 7, 10, 15, 20, 50\}$	[5]
γ	$\{1, 2, 3, 4, 5, 7, 10, 15, 20, 50\}$	[5]
ε	$\{1, 2, 3, 4, 5, 7, 10, 15, 20, 50\}$	[5]
PD	{0; 0,1; 0,2; 0,3; 0,4; 0,5; 0,6; 0,7; 0,8; 0,9; 1,0}	[0,5]
τ	{100, 500, 1000, 2000, 5000}	[2000]

Tabla 4.5: Parámetros y valores considerados en el proceso de sintonización.

El primer parámetro, $\beta \in [1, nodosDisponibles]$, indica el tamaño de la lista candidata a utilizar en el proceso de construcción. El parámetro controla la diversificación del proceso de construcción. Un valor dado de β sesga la solución a solo visitar la cantidad de nodos que β define y que además corresponden a los más cercanos al nodo actual, definiendo así la lista candidata como se aprecia en la linea 5 del algoritmo 3. Este valor determina cuantos nodos se utilizan al generar la ruleta. $\beta_{min} = 1$ selecciona como siguiente granja a visitar la más cercana al punto en que se encuentra el algoritmo. Con esto se obtiene un acercamiento mejor mejora, del ingles best-first search, lo cual construye siempre la misma solución inicial. Por otro lado, el valor más grande que puede tomar este parámetro varía en cada iteración pues corresponde a la cantidad de nodos disponibles para la inserción, entonces $\beta_{max} = nodosDisponibles$. Se utiliza $\beta = \beta_{max}$ cuando se inserta la primera granja a la ruta, para poder explorar diferentes secciones del espacio de búsqueda. Si este valor se mantiene durante toda la construcción, entonces se deja de restringir el tamaño de la lista candidata, con lo cual se pierde la consideración de la cercanía de las granjas.

El parámetro $\gamma \in [1, totalNodos]$ define la cantidad mínima de nodos a sacar de la solución para romper las demandas, con el fin de generar espacio suficiente para insertar nuevos nodos. El valor que tome γ siempre se ve condicionado por la cantidad de nodos que se deban sacar para lograr dejar de satisfacer las demandas. Por lo tanto, si se deben eliminar X nodos de la ruta para dejar de cumplir las demandas, cuando $X < \gamma$, entonces se debe continuar con la eliminación de nodos hasta eliminar γ nodos.

En el caso contrario, cuando $X \geq \gamma$ ya se habrán eliminado más de γ nodos, por lo que se

continua con el siguiente movimiento. Si $\gamma_{min} = 1$, se da la libertad de eliminar solo los nodos que sea necesario para dejar de cumplir con las demandas, ya que siempre se debe eliminar al menos uno. Por otro lado, $\gamma_{max} = totalNodos$ implica dejar todas las rutas con un solo nodo. Esto debido a la condición de la liena 3 del algoritmo 4.

Otro parámetro es $\varepsilon \in [1, \infty[$ y corresponde a la cantidad de nodos que se debe intentar agregar antes de decidir aumentar la temperatura del sistema bruscamente. La condición existente es agregar nodos hasta que no quepan más, sin importar la temperatura en que se encuentre el sistema. Entonces, si ε es muy grande y la temperatura es muy baja, se entra en un periodo de estancamiento. Esto se debe a que la temperatura permite insertar con mayor probabilidad solo los nodos que beneficien la función de evaluación y probablemente ya no queden nodos que cumplan esta condición. Es por esto que el valor de ε debe estar relacionado con la cantidad de nodos en la ruta donde se están intentando insertar los nodos, ya que ese valor corresponde a la cantidad de opciones de inserción que se tiene. En la línea 17 del algoritmo 5, este valor se encuentra multiplicando el tamaño de la ruta, con esto se busca intentar agregar en cada posición una determinada cantidad de veces un nodo. Por ejemplo, si $\varepsilon = 2$ entonces probablemente se intente insertar un nodo dos veces en cada posición. Con todo esto, si $\varepsilon_{min} = 1$ entonces cada vez que se rechaza la inserción de algún nodo *tamañoRuta* veces, se debe aumentar la temperatura. Dado que las rutas no son muy largas, se realiza anticipadamente el alza de temperatura, aceptando constantemente nodos que empeoren la función de evaluación y perdiendo un poco la idea de la meta-heurística Simulated Annealing.

Además, se tiene el parámetro $PD \in]0, 1[$ que se utiliza en el proceso de generar demandas ficticias. Como se explicó en la sección 3.2.7, se crea una ruleta en base a la multiplicación del total de leche producida de cada tipo por su ganancia correspondiente. Con un valor aleatorio se selecciona un tipo dentro de la ruta y se agrega $PD * capacidadDelCamion$ de ese tipo de leche a la demanda. Cada vez que se selecciona un tipo de leche, se debe armar de nuevo la ruleta dado que la cantidad de leche total disponible se ve modificada. De esta manera, se genera una demanda ficticia la cual es de vital importancia para las instancias que definen demandas nulas, ya que las demandas se consideran en varias partes del algoritmo propuesto. Si PD_{min} es cero no se crean demandas ficticias, porque se agrega cero litros de la

leche seleccionada en la ruleta. Por lo tanto, $PD_{min} > 0$.

Por otro lado, PD_{max} no puede ser uno, ya que eso implica demandar la totalidad de la capacidad de los camiones y probablemente no exista una configuración de granjas que produzcan exactamente dicha cantidad. Por lo tanto, $PD_{max} < 1$.

Un PD muy cercano a cero promueve saltos muy grandes en el espacio de búsqueda, pues la fase Romper Demandas tendrá que sacar muchos nodos de la ruta para dejar de cumplir las demandas. Por otro lado, un PD muy cercano a uno, no dejará lugar a visitar más granjas para sobre abastecer la planta. Un valor adecuado de PD, considera demandar una cantidad de leche que permita tener espacio disponible dentro del camión para sobre abastecer a la planta cuando es beneficioso.

El parámetro τ corresponde al número de iteraciones de mejora que se aplican a una misma solución inicial y μ son las veces que se reinicia la solución actual a una nueva solución inicial. Ambos parámetros se encuentran en la línea 2 y 4 del algoritmo 2. Se fijó un máximo de evaluaciones igual a 100000 y este corresponde a la multiplicación de τ y μ. Por lo tanto, es necesario solo configurar uno de estos valores considerando que el dominio de las variables se encuentra entre $[1, 100000]$. Al ser ambos parámetros inversamente proporcionales, un valor de τ muy bajo, implica un valor de μ alto. Dicha situación promueve la exploración del espacio de búsqueda, pues se aplican muchos reinicios, pero se mejora menos cada nueva solución inicial. Por el contrario, si τ es muy alto, se favorece la intensificación de la solución, lo cual significa que se aplican muchos movimientos de mejora a la solución actual antes de generar una nueva solución.

La temperatura inicial T_0 y α utilizadas en el algoritmo son parámetros que deben ser inicializados para este problema. Una regla de oro para inicializar adecuadamente la temperatura corresponde a generar un conjunto de soluciones iniciales s y seleccionar $T_0 \approx \sigma(f(s))..,2x\sigma(f(s))$, en este caso el valor obtenido corresponde a $T_0 = 500$. Por el lado del parámetro α, Laarhoven y Aarts [2] en 1988 recomiendaron valores de $\alpha \in [0{,}9 - 0{,}999]$. Para este estudio se utilizó un $\alpha = 0{,}99$.

A partir del proceso de sintonización se obtuvieron los siguientes valores para cada parámetro: $\beta = 15$, $\gamma = 3$, $\varepsilon = 10$, $PD = 0{,}7$ y $\tau = 500$.

Un valor $\beta = 15$ reduce la lista candidata del proceso de construcción a los 15 nodos más cercanos a la planta. Como se describe en la sección 3.2.7, los nodos que conforman la lista candidata se ven restringidos tanto por la calidad, como por la cantidad de leche producida. Considerando que las instancias poseen 40 nodos y tres tipos de leche, son pocas las ocasiones en que la lista candidata posea más de 15 nodos. Por esta razón, $\beta = 15$ le da la holgura a la construcción de elegir los nodos dentro de toda la lista candidata sin considerar su posición con respecto a la planta en la mayoría de las iteraciones.

Utilizar $\gamma = 3$ obliga a sacar al menos 3 nodos de la solución en el proceso de romper demandas detallado en la sección 3.2.8. Dependiendo del sobre abastecimiento que genera la solución y de la cantidad de leche que producen los nodos que se está eliminando, puede que un $\gamma = 3$ obligue a sacar más nodos. También puede ser que ya se estén eliminando más de 3 nodos y por lo tanto γ no aporte en nada. Con $\varepsilon = 10$ se intenta insertar muchas veces antes de recalentar, ya que este valor es multiplicado por los posibles lugares donde se puede insertar un nodo. Por lo tanto, $\varepsilon = 10$ busca que se intente realizar la inserción idealmente 10 veces por posición.

Cuando $PD = 0{,}7$ se demanda un 70 % de la capacidad total de la flota de camiones. Esto quiere decir que existe una 30 % de holgura para visitar más nodos y sobre abastecer a la planta. En ocasiones la holgura total es tan grande que puede superar la capacidad de un camión y por lo tanto pueden existir soluciones donde no se necesite utilizar la flota completa. Dado que una de las restricciones del problema es asignar una ruta a cada vehículo, el camión tendrá que recoger leche para sobre abastecer la granja de la mejor forma que pueda.

Un $\tau = 500$ genera un $\mu = 200$ por lo que se tendrán 200 reinicios y 500 iteraciones de mejora, favoreciendo la intensificación.

4.1.4. Entorno experimental

Cada una de las pruebas de este trabajo de memoria fue ejecutada con 20 semillas diferentes de manera de lidiar con 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.

4.2. Resultados

En esta sección se presentan los resultados obtenidos durante el proceso de evaluación del algoritmo propuesto. Primero se presentan los resultados asociados al análisis de componentes, a continuación se presentan los resultados de la comparación con un enfoque de resolución completa y finalmente se presentan los resultados de la evaluación con instancias reales.

4.2.1. Análisis de componentes

A continuación se analiza la relevancia de los parámetros del algoritmo en la calidad de los resultados obtenidos en las distintas instancias. Para está comparación se considera solo el conjunto de instancias pequeñas y los resultados obtenidos se presentan utilizando boxplots.

En cada uno de los gráficos presentados en las siguientes secciones, el boxplot central es de color rosado y siempre corresponde al valor que toma el parámetro tras la sintonización. Por su parte, el boxplot celeste se encuentra a la izquierda y siempre considera un valor para el parámetro menor que el valor utilizado y el boxplot de la derecha de color verde un valor mayor.

Se presenta el análisis de los parámetros β que controla el largo de la lista candidata del proceso de construcción, γ que controla la cantidad de nodos que se eliminan de las soluciones en el movimiento Romper demandas, ε que controla la cantidad de intentos de inserción de nodos en el movimiento Agregar nodos, PD que controla el porcentaje de demanda ficticia en las instancias sin demanda y τ que controla la cantidad de iteraciones de mejora.

Largo de la lista candidata en Construcción

Como se ha explicado anteriormente, el parámetro que limita el largo de la lista candidata utilizada en el proceso de construcción corresponde a β. El dominio de este parámetro es $\beta \in [1, nodosDisponibles]$.

Se realizaron experimentos con $\beta = 1$, boxplots de color verde, lo cual genera un acercamiento mejor mejora al reducir la lista candidata al nodo más cercano. En los bloxplots de color azul se utilizó un $\beta = 40$, es decir, no cortar la lista candidata y por tanto, no tomar en consideración la distancia entre los nodos. Lo anterior se debe a que la cantidad de nodos de todas las instancias es 40 siendo imposible tener una lista mayor a este valor. El valor obtenido para β tras la sintonización corresponde a $\beta = 15$, donde si se considera cortar la lista candidata, dejando fuera los nodos más alejados. Como es posible apreciar en la figura 4.5 los experimentos realizados con un $\beta = 1$ no entregaron buenos resultados ya que las medias de los boxplots celestes suelen encontrarse por debajo de las medias del parámetro sintonizado. La excepción se encuentra en la cuarta instancia, donde $\beta = 1$ es mejor que $\beta = 15$. La tabla 4.2 indica que la instancia 4 no posee demandas, por lo que los resultados obtenidos se ven estrechamente relacionados con las demandas ficticias creadas. En las instancias 2, 4 y 5 se logra obtener un buen máximo cuando $\beta = 1$, pero en todas ellas se trata de un valor atípico. Estas tres instancias poseen en común la flota de camiones conformada por 5 vehículos con tanques de $30000L$.

En el otro extremo, con $\beta = 40$ se obtuvieron resultados muy similares en relación al valor del parámetro una vez sintonizado. Esto se debe a que, como se explicó anteriormente, existen pocas ocasiones donde la lista candidata sea mayor a 15 nodos, por lo que el parámetro β no suele aplicarse.

Solo en la instancia 5 no cortar la lista candidata, boxplot verde, tiene una media y valor máximo considerablemente mejor que $\beta = 15$. Esto se debe a la forma de construir las soluciones iniciales que se utiliza en el algoritmo 3 y las características de la instancia. En esta instancia se logra tener listas candidatas mayores a 15 nodos utilizando el parámetro β con mayor frecuencia dejando de considerar los nodos lejanos. La instancia demanda $10000L$ de leche tipo A, es decir, solo un 16 % del total producido de ese tipo. Para suplir esta demanda

se necesitan visitar pocos nodos, ya que como se presenta en la figura 4.2 más de un 50 % de los nodos producen más de 2500L de leche tipo A. Por lo tanto, cuando el algoritmo pasa por la linea 8 y empieza una nueva ruta, la lista candidata contempla los 13 nodos de tipo B más todos los que no fueron visitados en la ruta anterior, ya que todos ellos producen menos litros que los 30000L de capacidad que posee el camión.

Se sigue que las siguientes iteraciones del algoritmo de construcción también implican más de 15 nodos, utilizando el corte con mayor frecuencia. Como se ha explicado anteriormente, cortar la lista candidata implica dejar fuera los nodos más lejanos. Como se muestra en la tabla 4.1, el nodo 40 se encuentra bastante aislado del resto de los nodos. Al encontrarse tan lejos de la planta, este nodo queda fuera cuando se aplica un corte con $\beta = 15$ y en las siguientes iteraciones tampoco sera incluido pues su producción, de 20000L, supera la capacidad del camión. Por lo tanto, para la quinta instancia con $\beta = 15$ el nodo 40 no se encuentra dentro de la solución y es justamente ese nodo el que sí aparece en las soluciones que no consideran corte. Todo esto justifica la distribución y la mediana de la instancia 5.

En cuanto a la dispersión de los datos, en la instancia tres todos los valores del parámetro β entregan resultados muy cercanos entre sí. Para $\beta = 1$ los resultados obtenidos son tan similares que solo es posible apreciar la linea de la mediana. Para los otros dos valores de β el 50 % de los resultados son equivalentes, por eso no se aprecia ni el mínimo ni el primer cuartil, pues corresponden al mismo valor.

Para la sexta instancia, el algoritmo propuesto no fue capaz de construir una solución inicial cuando $\beta = 1$ por lo tanto para generar el gráfico y analizar resultados se utilizó un $\beta = 2$. Con este valor si fue posible obtener resultados y estos se encuentran muy poco dispersos y poseen una mediana levemente por debajo de las obtenidas por los otros valores de β para esa instancia.

Considerando los tiempos promedios de ejecución de cada instancia con los diferentes valores para el parámetro β, podemos ver en el gráfico de la figura 4.4 que el mayor aumento ocurre en la quinta instancia la cual es justamente la que posee la mayor dispersión de los datos. Al tener un camión libre para recolectar la leche que sobre abastece a la planta, un beta=40 le permite moverse hacia donde sea sin considerar lo lejos que esto podría estar. Por

lo tanto se explora bastante el espacio de búsqueda obteniendo rutas muy variadas entre las diferentes soluciones.

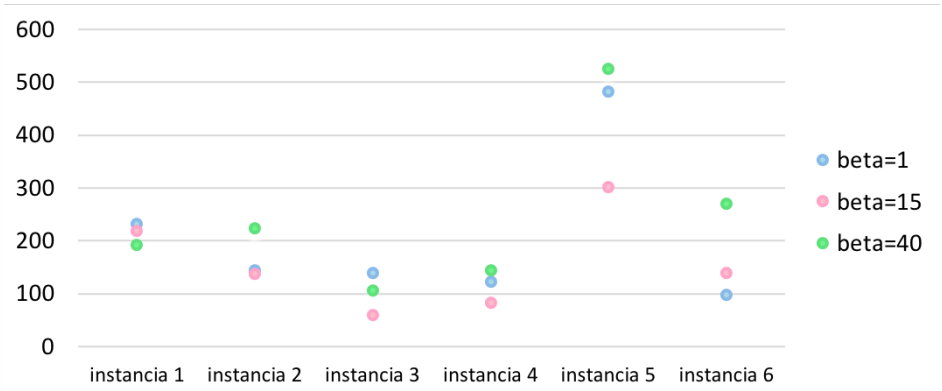
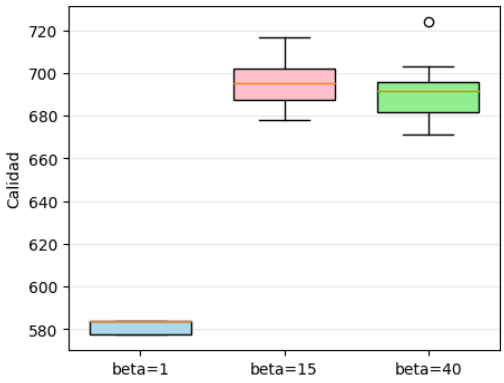
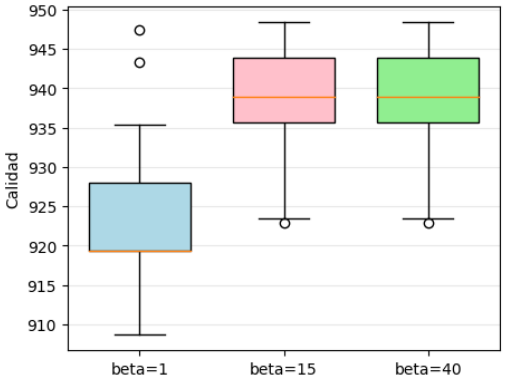


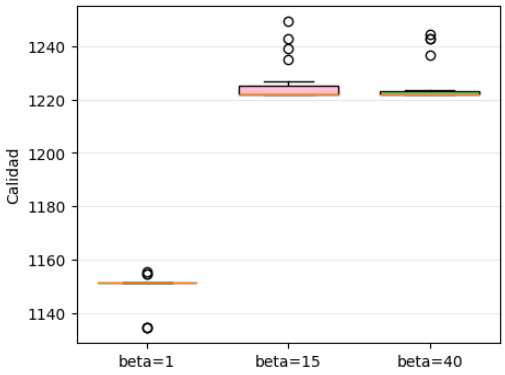
Figura 4.4: Comparación de los tiempos promedio de ejecución del algoritmo variando el parámetro β.



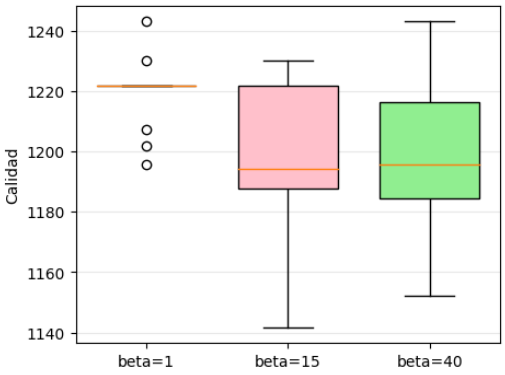
Instancia 1



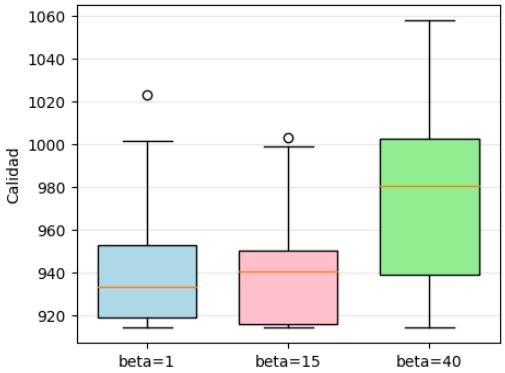
Instancia 2



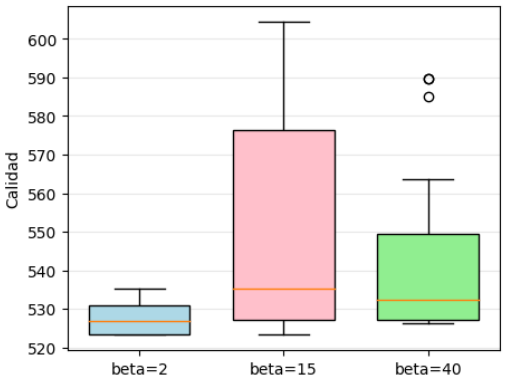
Instancia 3



Instancia 4



Instancia 5



Instancia 6

Figura 4.5: Calidad de soluciones variando el parámetro β.

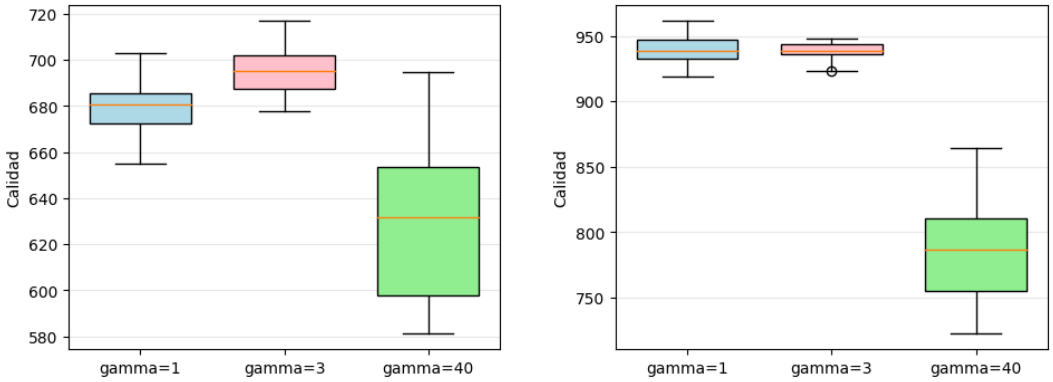
Número de nodos a eliminar en Romper Demandas

En el algoritmo 4 se utiliza el parámetro γ para determinar la cantidad mínima de nodos que se deben eliminar. Su dominio es $\gamma \in [1, 40]$ y se realizaron pruebas con los valores extremos $\gamma_{min} = 1$ en celeste y $\gamma_{max} = 40$ en verde. La primera opción deja a criterio del sobre abastecimiento la cantidad de nodos a eliminar. La segunda opción implica reducir las rutas a un solo nodo, como se aprecia en la linea 3 del algoritmo 4. La sintonización seleccionó un $\gamma = 3$ obligando a la fase Romper Demandas a sacar al menos tres nodos de sus rutas antes de continuar con la siguiente fase.

En la figura 4.6 se presentan los resultados obtenidos en está experimentación. Al comparar $\gamma = 3$ con $\gamma = 1$ se esta comparando un algoritmo que obliga a sacar una cierta cantidad de nodos con uno que no. La idea de forzar la eliminación es crear un mayor espacio dentro de las rutas y así permitir que se agreguen nuevos nodos favoreciendo la exploración del espacio de búsqueda. Boxplots similares, como ocurre en la mayoría de las instancias, se producen pues en la mayoría de las iteraciones, fue necesario eliminar los tres nodos para poder dejar de cumplir con las demandas.

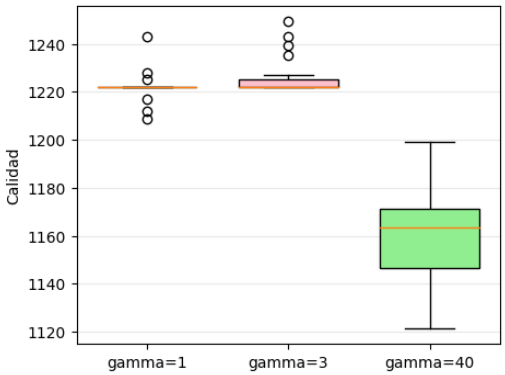
En la sexta instancia, como se ha mencionado anteriormente, los 90 litros de leche demandada tendrán que ser cubiertos por leche de tipo A y B. Además, en un caso ideal, se posee solo un 10 % de la capacidad total de trasporte disponible para sobre abastecimiento de la planta. Dada esta situación, cuando se elimina un nodo es bastante probable que inmediatamente se dejen de suplir las demandas y que no existan muchos otros nodos disponibles para insertarse en la siguiente fase, por esta razón la distribución de los resultados para $\gamma = 1$ en la instancia 6 se encuentra tan concentrada.

Las soluciones obtenidas utilizando un γ_{max} no son muy favorables. En la gran mayoría se posee una gran dispersión de los datos y una media bastante baja en comparación a los otros valores de gamma. Esto se debe precisamente al hecho de que $\gamma = 40$ destruye todo el trabajo realizado hasta ese punto. Por lo tanto, la siguiente fase, Agregar Nodos descrita en el algoritmo 5, se debe encargar de crear una nueva solución a partir de rutas con solo dos nodos. Esto lleva a diversificar mucho, pero intensificar muy poco, dado que se pierden las mejoras aplicadas a la solución al volver a crearla.

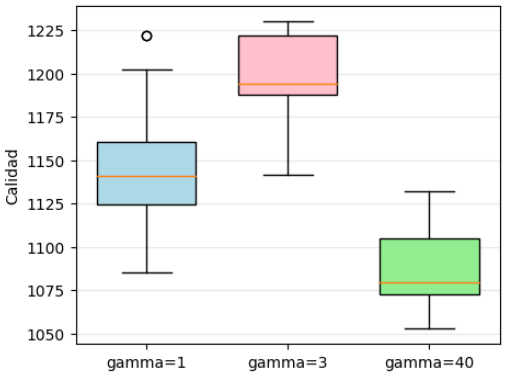


Instancia 1

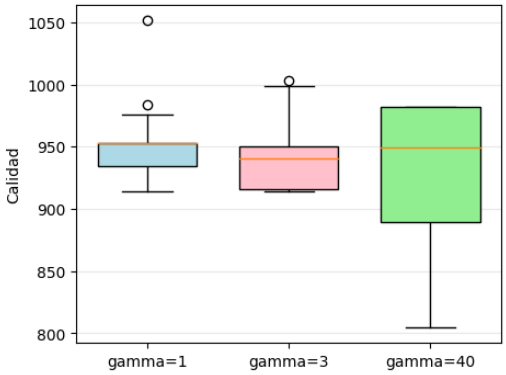
Instancia 2



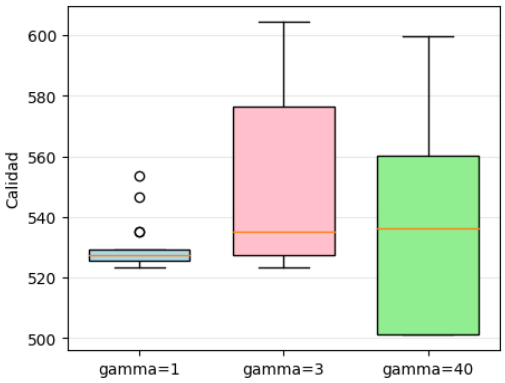
Instancia 3



Instancia 4



Instancia 5



Instancia 6

Figura 4.6: Comparación calidad de soluciones variando el parámetro γ.

En la quinta instancia nuevamente se posee un tiempo muy grande en comparación al resto para *gamma* = 40 como se aprecia en el gráfico 4.7. En dicho gráfico no aparece el tiempo de ejecución para la instancia 2 cuando *gamma* = 40, ya que en esa situación se excede el

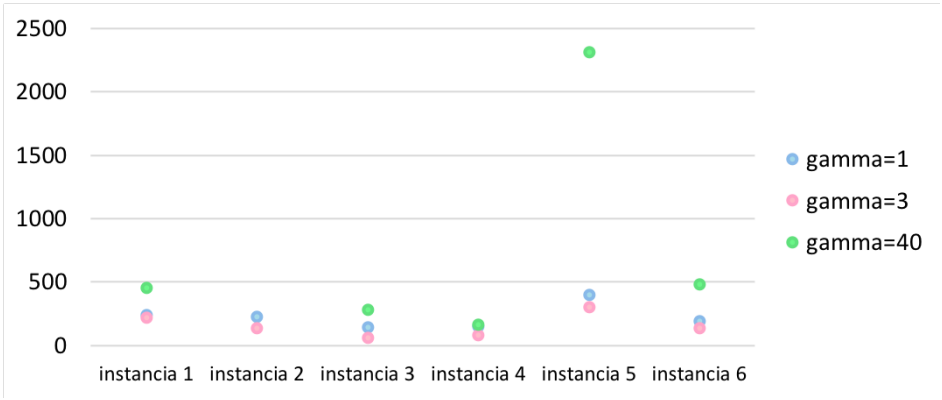


Figura 4.7: Tiempos promedio de ejecución del algoritmo variando el parámetro γ.

tiempo limite establecido de 32767 segundos. Esto se produce porque a la fase de Agregar Nodos le debe costar muchas iteraciones lograr obtener nuevamente una solución factible a partir de las rutas con un solo nodo generadas por el valor que toma γ. Incluso en algunas iteraciones no debe lograr obtener la nueva solución factible y se queda iterando por más del tiempo establecido.

Intentos de inserción de nodos en Agregar Nodos

El parámetro que se encuentra relacionado con la meta-heurística Simulated Annealign corresponde a ε. Este parámetro se utiliza en la fase Agregar Nodos descrita en el algoritmo 5. El dominio es $\varepsilon \in [1, \infty[$.

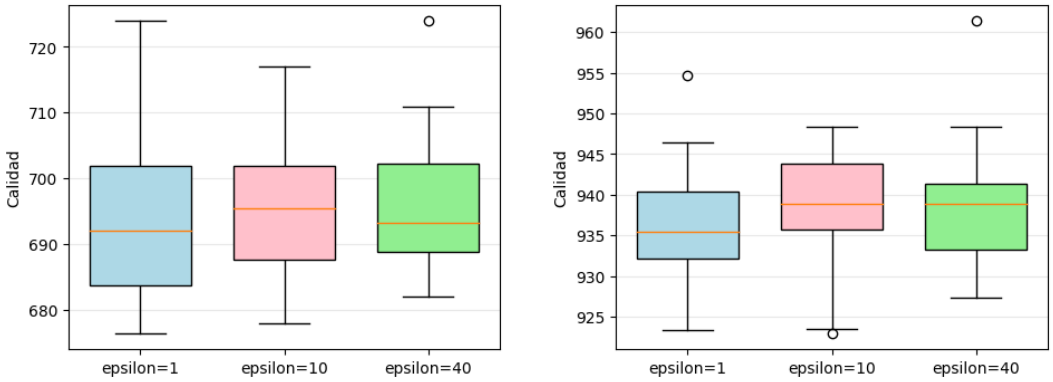
Las pruebas ejecutadas utilizan $\varepsilon_{min} = 1$ representadas en boxplots celestes y $\varepsilon = 40$ en color verde. ε_{min} implica recalentar cuando se rechaza *tamañoRuta* veces el intento de insertar un nodo en dicha ruta. Esto genera que cuando la temperatura comienza a bajar y dado que las rutas no suelen contener muchos nodos, se va a recalentar el sistema de manera anticipada sin dejar que la meta-heurísica se desempeñe adecuadamente. La temperatura siempre estaría alta y constantemente sería posible admitir movimientos que empeoren la función de evaluación. Básicamente se estropearía la meta-heurística utilizada, pero esto no sucede tan agresivamente gracias a la fase Remover Nodos, que posee el algoritmo implementado. En esta fase, como se explica en la sección 9, se sacan de la solución todos aquellos nodos que

resulten perjudiciales para la función de evaluación. Por otro lado, $\varepsilon = 40$ no permite recalentar la temperatura del sistema si no hasta que se haya intentado insertar idealmente unas 40 veces algún nodo en cada trayecto de la ruta. La sintonización arrojó que un buen valor para ε considerando todas las instancias corresponde a $\varepsilon = 10$.

En la figura 4.8 se presentan los gráficos asociados a los resultados obtenidos tras la experimentación descrita. Como es posible apreciar, las medianas de todos los boxplots en cada gráfico son bastante similares, lo que varia entre uno y otro es la dispersión de los resultados. Con $\varepsilon = 1$, dado que la temperatura se suele mantener alta, se acepta insertar tanto los nodos que beneficien a la función objetivo como los que no. De esta manera, al tener más opciones de nodos a insertar, se realiza una mayor exploración del espacio de búsqueda. Con esto se obtienen soluciones con rutas más diversas y por ende la dispersión de los resultados es más amplia. Esto se aprecia claramente en las instancias uno, dos y cuatro, donde a medida que aumenta $\varepsilon = 1$, las soluciones son menos dispersas.

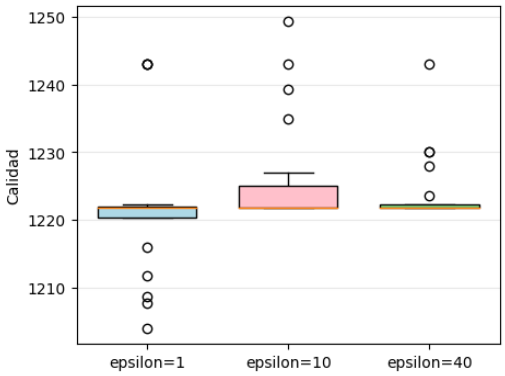
Por su parte, el comportamiento del boxplot celeste de la instancia seis, está relacionado con el análisis del mismo boxplot realizado en la sección anterior. En resumen, debido a las características de la instancia se tienen pocas opciones para insertar. Esto hace que el boxplot quede más condensado, porque el algoritmo juega siempre con los mismos nodos y la temperatura obliga a seleccionar aquellos que beneficien a la función objetivo.

Finalmente, la instancia 5, como se ha dicho anteriormente, es la que posee mayor holgura en cuanto a espacio para transportar leche que sobre abastecerá a la planta. Una vez se repara la demanda, el algoritmo 5 tendrá harto espacio disponible para seguir insertando nodos y explorar el espacio de búsqueda. Al utilizar una temperatura baja, se seleccionaran solo nodos que beneficien la función de evaluación y que generen mejores soluciones.

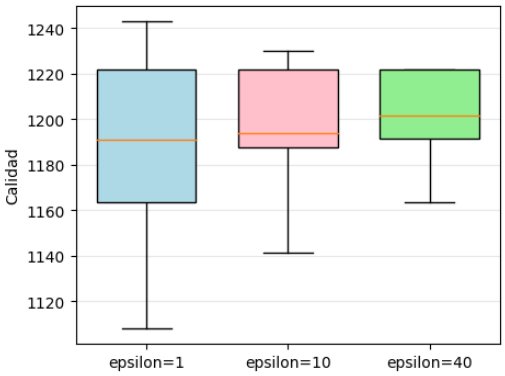


Instancia 1

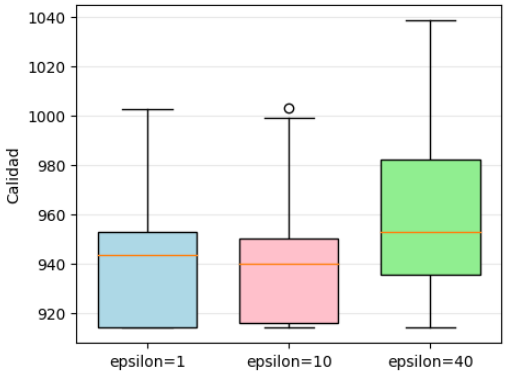
Instancia 2



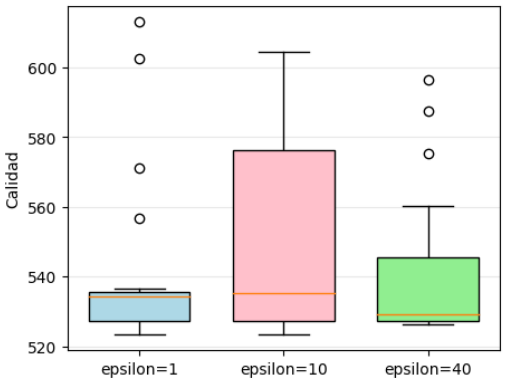
Instancia 3



Instancia 4



Instancia 5



Instancia 6

Figura 4.8: Calidad de soluciones variando el parámetro ε.

Al mirar los tiempos promedios de ejecución en el gráfico 4.9 no hay ninguna instancia que posea una diferencia muy relevante en comparación al tiempo obtenido utilizando el parámetro sintonizado.

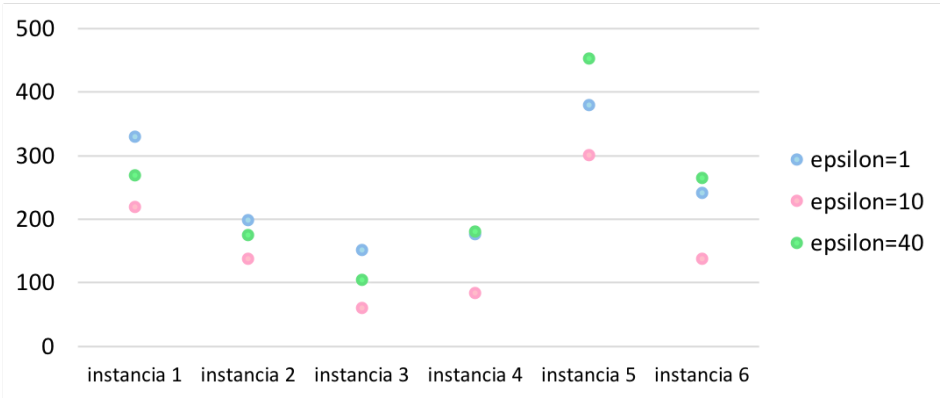


Figura 4.9: Tiempos promedio de ejecución del algoritmo variando el parámetro ϵ.

Porcentaje de demanda ficticia en Construcción

El parámetro *PD* hace sentido solo cuando se deben crear demandas ficticias, pues no se utiliza cuando la planta ya demanda ciertas cantidades de leche. *PD* pertenece a los números racionales y toma un valor entre $]0, 1[$ dado que representa qué porcentaje de la capacidad de los camiones va a ser demandado por la planta.

Los experimentos realizados consideraron por cada instancia un $PD = 0{,}6$ ilustrado en color celeste y un $PD = 0{,}8$ en verde. Se tomaron estos valores tras analizar las cuatro instancias que si tienen demandas. El menor valor seleccionado para las pruebas corresponde a $PD = 0{,}6$. Con este valor se genera porcentualmente la misma disponibilidad de estanque para sobre abastecer la planta que la instancia 5, la cual es la que más holgura entrega.

Por otro lado, considerando que las instancias sin demandas deben llevar la cantidad de leche que estimen beneficiosa para la función objetivo, se decidió comparar con un $PD = 0{,}8$ para dejar algo de espacio libre dentro de los camiones.

Solo las instancias 3 y 4 no poseen demandas generando diferentes resultados para los distintos valores de *PD*. El resto de las instancias no hizo uso de este parámetro y por esta razón todos los boxplots son iguales.

A partir del parámetro *PD* y las características de las instancias 3 y 4 se generó la tabla 4.6. Todos los valores bajo la columna *Capacidad* se encuentran expresados en miles de litros. En

esta tabla se puede observar en cuanto cambia la capacidad total utilizada con los diferentes valores de *PD*. Como se puede ver en la columna *cam.* la diferencia entre ambas instancias corresponde a un solo camión, pero esto es suficiente para generar una diferencia significativa en las capacidades utilizadas y disponibles. Cabe mencionar, que la columna *Capacidad utilizada* además indica el total de litros de la demanda ficticia, ya que ésta se construye a partir de la flota de camiones como se explica en la sección 3.2.7. Debido a las diferentes producciones de los nodos, consultar la tabla 3.1 del anexo, no es posible recolectar siempre la cantidad exacta demandada, pero esta columna permite tener una idea de cuántos litros se poseen de holgura para transportar leche que sobre abastecerá a la planta.

			Capacidad			
Instancias	**PD**	**# cam.**	Por Camión	Total	Utilizada	Disponible
instancia 3	0.6	4	30	120	72	48
instancia 3	0.7	4	30	120	84	36
instancia 3	0.8	4	30	120	96	24
instancia 4	0.6	5	30	150	90	60
instancia 4	0.7	5	30	150	105	45
instancia 4	0.8	5	30	150	120	30

Tabla 4.6: Demandas ficticias instancias 3 y 4 con $PD = [0{,}6; 0{,}7; 0{,}8]$.

Considerando la forma en que se crean las demandas ficticias como máximo se va a demandar un 60 %, 70 % u 80 % del total producido por cada tipo de leche. Así, nunca se obtendrá una configuración de demandas donde se pida más del total existente de cierto tipo de leche como ocurre en la quinta instancia. Con esto se busca evitar crear alguna configuración no factible de demandas. Además, dado que realmente las instancias no poseen demandas y dejan a criterio del algoritmo que nodos visitar, no parece razonable obligar a la instancia a recolectar la totalidad de ninguno de los tipos de leche.

La dispersión de los datos en la tercera instancia es bastante pequeña, el efecto visual producido por la escala del eje *Y* los hace parecer muy dispersos. De hecho, estos se encuentran más agrupados que los boxplots de la instancia 4, aún cuando en esta última se ven las cajas más pequeñas.

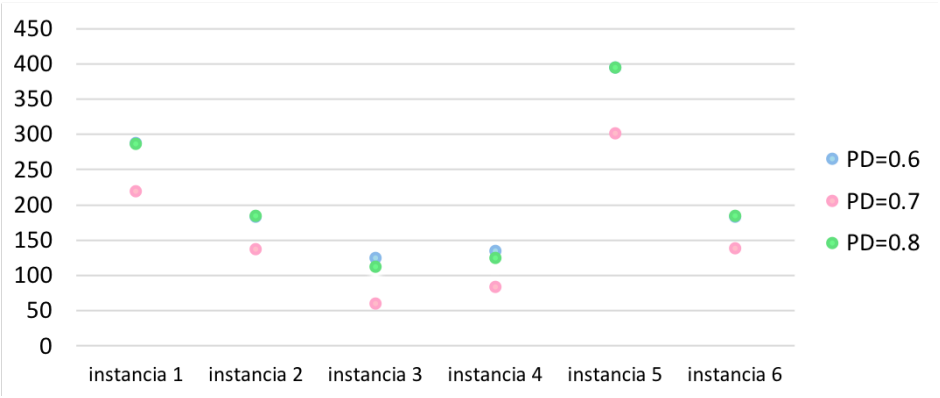
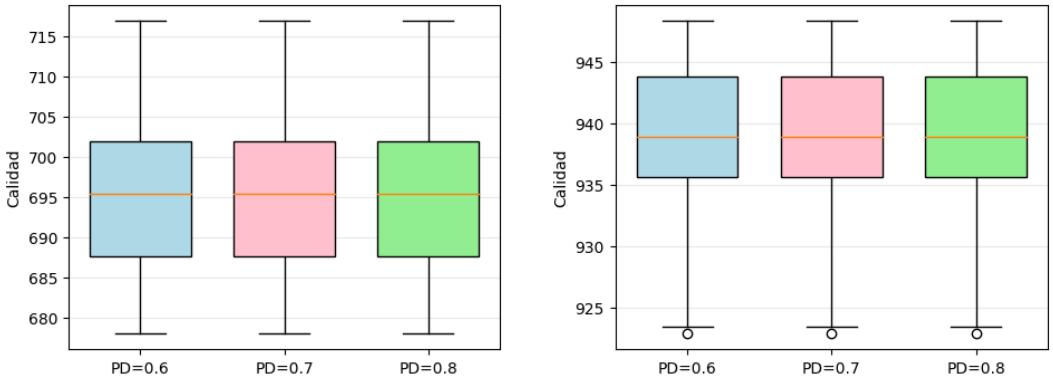


Figura 4.10: Tiempos promedio de ejecución del algoritmo variando el parámetro *PD*.

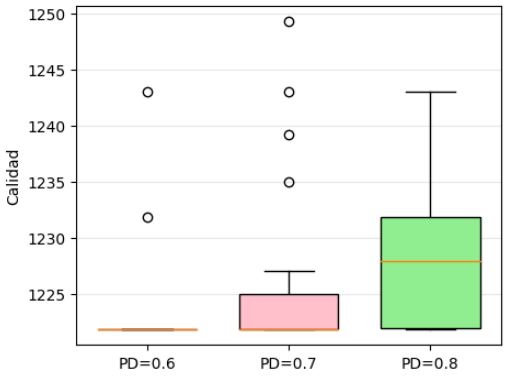
En la tercera instancia un $PD = 0{,}8$ obtiene mejores resultados que el resto, pero al utilizar este mismo valor de *PD* en la instancia 4 se tienen peores resultados como se puede observar en la figura 4.11. Por otro lado, un $PD = 0{,}6$ obtiene datos muy poco dispersos, especialmente en la tercera instancia donde obtiene el mismo resultado para casi todas las semillas.

En cuanto a los tiempos de ejecución promedio de las instancias presentadas en el gráfico 4.10 podemos apreciar que para todas las instancias existe un aumento en comparación al parámetro sintonizado. Esto ocurre debido a que la creación de demanda ficticia se realiza siempre, pero no se utilizan cuando ya existen demandas por parte de planta. Con esto se busca que el tiempo empelado en la creación de demandas ficticias no se adjudique a las diferencias temporales que puedan tener este tipo de instancias con el resto.

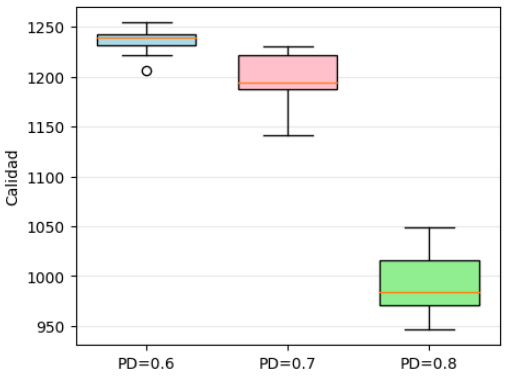


Instancia 1

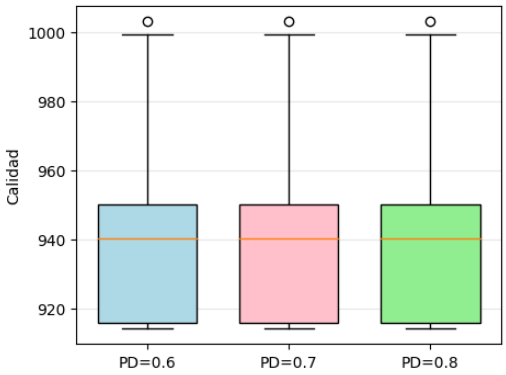
Instancia 2



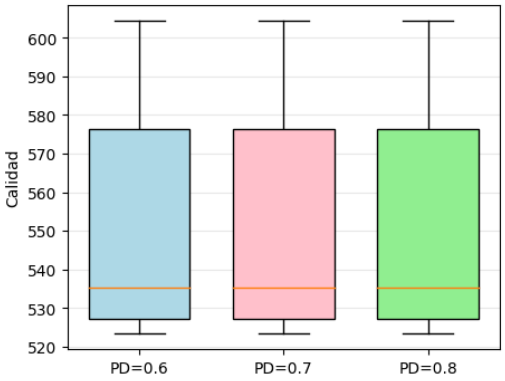
Instancia 3



Instancia 4



Instancia 5



Instancia 6

Figura 4.11: Calidad de soluciones variando el parámetro *PD*.

Número de iteraciones de mejora

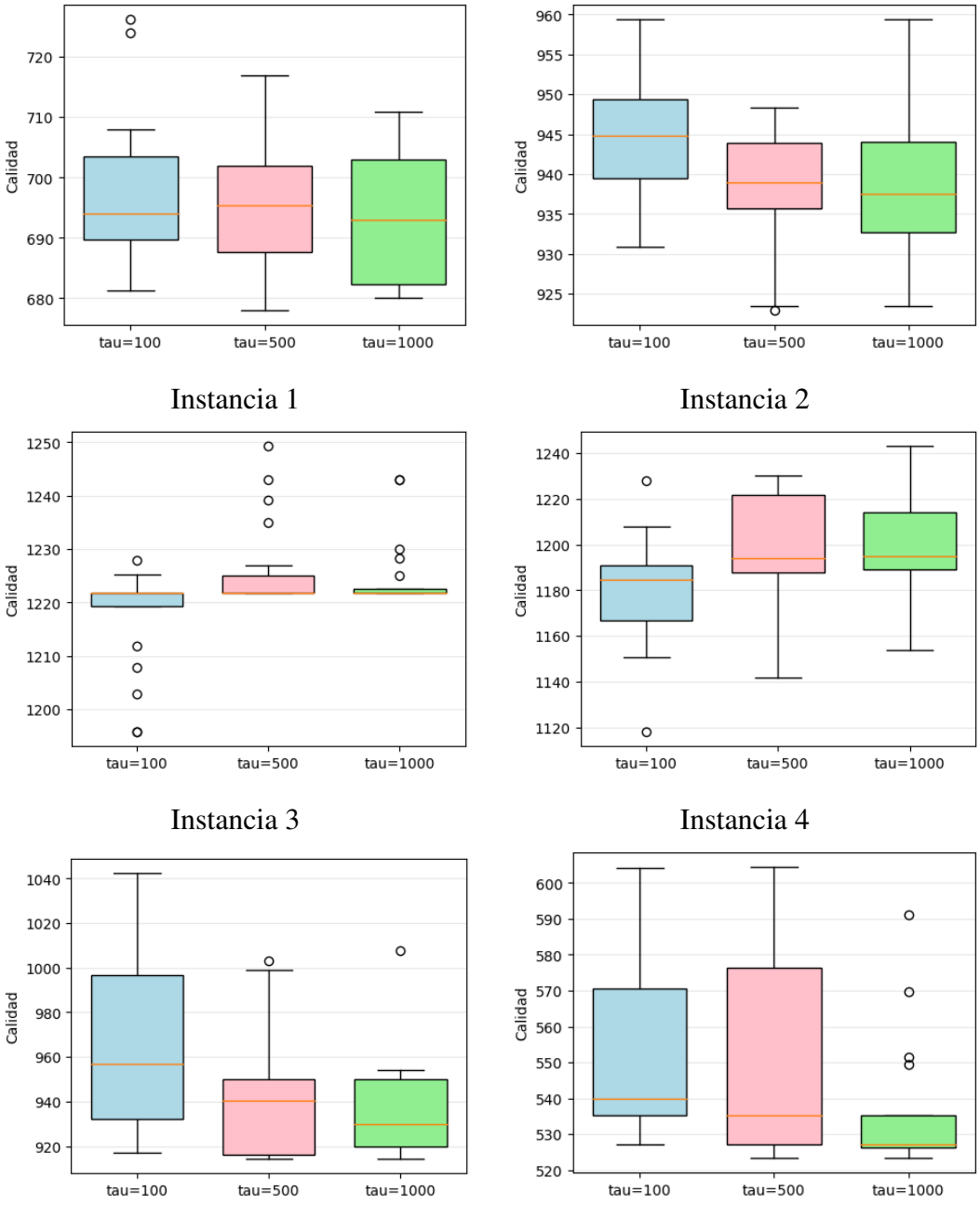
El número de iteraciones de las fases de mejora τ esta inversamente relacionado con el número de reinicios μ que tendrá el algoritmo. Dado $\mu = \frac{100000}{\tau}$ se posee un dominio para $\tau \in [1, 100000]$.

Las pruebas realizadas contemplan un $\tau = 100$ en los boxplots celestes, con lo cual se obtiene un $\mu = 1000$ y la otra opción analizada es la contraria, un $\tau = 1000$ con $\mu = 100$ ilustrado en verde. El valor obtenido para τ tras la sintonización corresponde a 500 por lo que $\mu = 200$.

En la figura 4.12 se presentan las visualizaciones relacionadas a los resultados encontrados con los valores ya descritos de τ. Las medianas de las primeras cuatro instancias son bastante similares entre sus boxplots, teniendo una diferencia máxima de $5u.d.$. Las dispersiones de los primeras dos gráficos son bastante similares encontrándose en un rango de aproximadamente $30u.d.$. En la última instancia la dispersión es más grande que en los otros gráficos pero sus medianas siguen estando bastante cercanas.

Para la quinta instancia se poseen resultados más variados. Al utilizar un $\tau = 100$ obtiene mejores resultados, pero estos se encuentra más dispersos. Utilizar este valor de τ implica considerar un μ igual a 1000, con lo cual se prioriza la diversificación generando un mayor número de soluciones iniciales. Probablemente, esta mayor varianza se deba a que en esta instancia prácticamente hay un camión de holgura, es decir, luego de construir la ruta de 4 camiones, ya se logra suplir la demanda de la planta y por lo tanto el ultimo camión puede ir a visitar cualquier nodo. Se sigue que si se construye muchas veces, este camión se moverá libremente por el espacio de búsqueda obteniendo rutas muy diversas y, por lo tanto, resultados dispersos. Esto mismo explica el aumento en el tiempo de ejecución promedio que tiene esta instancia para el valor sintonizado y $\tau = 10000$ en comparación al resto de los tiempos, lo que se aprecia en la figura 4.13.

La similitud en los boxplots de cada gráfico en las instancias 3 y 4, que no poseen demandas, indica que la forma de crear las demandas ficticias es adecuada. Dado que sin importar el tamaño de la muestra, es decir cuántas demandas ficticias se creen, ésta es representativa.



Instancia 1 Instancia 2

Instancia 3 Instancia 4

Instancia 5 Instancia 6

Figura 4.12: Calidad de soluciones variando el parámetro τ.

Finalmente, viendo la gráfica de la figura 4.13 nuevamente es la quinta instancia la que presenta una diferencia significativa, pero en este caso, para $\tau = 1000$ el tiempo es mucho menor que para los otros valores del parámetro. Con dicho valor de τ se prioriza la intensificación,

correspondiente a los pasos entre las lineas 5 y 19 del algoritmo 2. Probablemente esta disminución considerable en el tiempo promedio de ejecución de esta instancia se deba a que generalmente el algoritmo 2 no encuentra una solución factible en la linea 9 y por lo tanto se ahorra el tiempo de ejecución de las fases Eliminar Nodos y Reordenar Nodos ya que no las realiza.

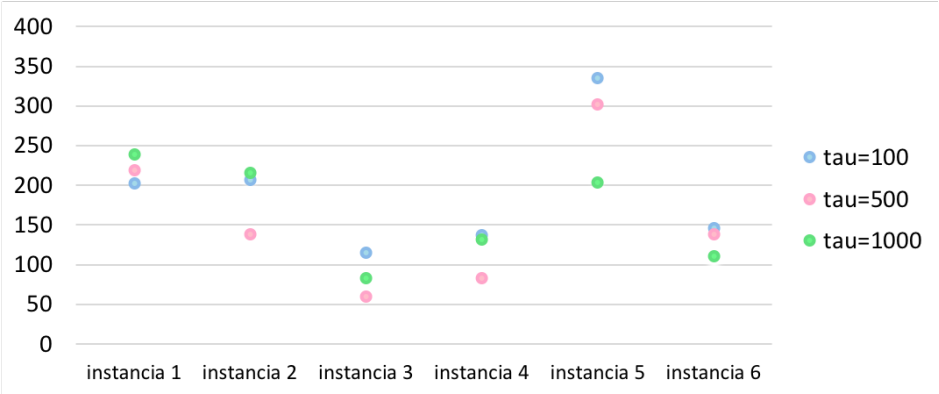


Figura 4.13: Tiempos promedio de ejecución del algoritmo variando el parámetro τ.

4.2.2. Conclusiones del análisis de componentes

Los valores sintonizados corresponden a los que presentan mejores resultados en general para el conjunto de instancias. Puede que en ciertas instancias un valor diferente de un parámetro funcione mucho mejor que el sintonizado, pero, al considerar el conjunto de instancias completo, el valor sintonizado es el mejor.

El análisis de los componentes demuestra que es bueno generar una lista candidata y utilizar un corte en vez de realizar una implementación mejor mejora, justificando la existencia del parámetro β. También se concluye que obligar a sacar un cierto número de nodos es mejor que sacar solo los necesarios para romper las demandas por lo que γ también se justifica. El parámetro ε es de suma importancia para el algoritmo pues establece el momento en que la temperatura se debe subir bruscamente, seleccionar un valor adecuado para este parámetro implica no realizar iteraciones innecesarias y por tanto reducir los tiempos de ejecución. PD es indispensable para la creación de demandas ficticias, sin este parámetro no se podría

resolver las instancias sin demandas. Finalmente, los parámetros μ y τ, por lo general, se encuentran presentes en casi todas las meta-heurísticas y su valor está relacionado con cuántas veces o cuánto tiempo se desea ejecutar el algoritmo y en esta ocasión sirvieron para evaluar el conjunto de demandas ficticias generadas.

4.2.3. Comparación con acercamiento completo

En esta sección se presentan los resultados comparando las soluciones obtenidas utilizando el algoritmo propuesto con los que se puede obtener utilizando un algoritmo completo. Para esto se utilizó el solver CPLEX para resolver el modelo de programación entera asociadas a las instancias pequeñas. Estos resultados fueron obtenidos después de 30 minutos en cada una de las instancias.

La tabla 4.7 muestra los resultados obtenidos por el acercamiento completo comparados con los mejores resultados del acercamiento propuesto para cada una de las instancias de prueba. Se indica la instancia, la ganancia asociada a cada mejor solución, los costos de transporte de dicha solución y el total. Además, se indica el gap porcentual, o diferencia porcentual, comparando los resultados de ambos acercamientos.

Instancia	**CPLEX**			**Simulated Annealing**			
	Ingreso	Costo	Total	Ingreso	Costo	Total	GAP
Instancia 1	2768.3	2018.0	750.3	2784.9	2068.0	716.9	-4.4
Instancia 2	2803.4	1834.0	969.4	2803.4	1855.0	948.4	-2.2
Instancia 3	2877.3	1623.0	1254.3	2770.3	1521.0	1249.3	-0.4
Instancia 4	2976.0	1665.0	1311.0	2866.0	1636.0	1230.0	-6.2
Instancia 5	10047.9	8807.3	1240.6	2730.0	1727.0	1003.0	-19.1
Instancia 6	2294.1	1690.0	604.1	2268.4	1664.0	604.4	0.0

Tabla 4.7: Calidad de la propuesta versus modelo de programación lineal entera.

Para la primera instancia se posee un GAP de 4,4 %, el algoritmo basado en Simulated Annealing encuentra una solución que posee un mayor ingreso, pero también un mayor costo

de transporte con lo cual se obtiene una solución 33$u.m.$ más baja que la encontrada por CPLEX.

En la segunda instancia, donde se obtiene un gap de 2,2 %, el ingreso es el mismo en ambas soluciones, pero se difiere en el costo, por lo tanto el CPLEX encuentra una ruta que es más corta que la encontrada por el SA. Esto también puede suceder cuando un nodo se encuentra en la ruta de otro camión. El algoritmo propuesto puede ser capaz de encontrar dicha solución si en la fase Romper Demandas se saca este nodo y luego en la siguiente fase, Agregrar Nodos, se inserta en la ruta que corresponde.

El algoritmo propuesto obtuvo una solución muy cercana a la obtenida por el CEPLEX para la instancia 3, donde el gap es de un 0,4 %. Aún cuando el gap es muy pequeño, si se comparan los ingresos y costos obtenidos en ambos casos se puede apreciar que las dos soluciones deben contener rutas bastante diferentes. Tras analizar la tabla 4.11 del anexo que describe la matriz de distancias, se puede decir que existe la posibilidad de que esta diferencia tan alta en los costos e ingresos se produzca por la modificación de un solo nodo. La diferencia existente entre los costos implica que la solución del algoritmo SA recorre 102$u.d$. más, valor que no es muy alto y se puede obtener involucrando un solo nodo.

Las instancias cuatro y cinco son las que obtienen el peor desempeño en comparación al acercamiento completo. La primera posee un gap del 6,2 % y la segunda uno casi el triple más grande equivalente a 19,1 %. A modo de recuerdo, estas dos instancias poseen una flota de 5 camiones con capacidad de 30000L cada uno. Como se muestra en la tabla 4.2, la instancia 5 posee una holgura del 40 % de la capacidad total de su flota y la instancia 4 no posee demandas por lo que se deben crear demandas ficticias. Considerando que el parámetro PD tras la sintonización obtuvo el valor 0,7, entonces la cuarta instancia tendrá una holgura del 30 %. Los resultados indican que el algoritmo propuesto se comporta mejor cuando se tiene menor holgura, es decir, cuando se obliga a recolectar una cantidad de leche que implica ocupar mayoritariamente la capacidad de transporte total de la flota, dejando poco espacio para sobre abastecer la planta.

El mejor resultado se obtiene en la instancia 6, donde se encuentra una solución que es levemente mejor que la obtenida por el acercamiento completo. Los ingresos obtenidos por

el CPLEX son 25,7$u.m.$ mayores que los obtenidos por el Simulated Annealing, pero el costo de transporte es 26$u.m.$ menor en el SA. Por lo tanto, se tiene una diferencia de 0,3$u.m.$ Debido a la existencia de diferencias tanto en los costos como en los ingresos, probablemente la diferencia entre ambas soluciones corresponda a visitar un conjunto diferente de nodos o que cierto nodo es visitado por un camión que recolecte otro tipo de leche cambiando su ingreso.

Es importante notar, que en la creación de las instancias reales se utilizaron los porcentajes demandados por las instancias pequeñas con peor desempeño, correspondientes a las demandas de la instancia 4 y 5, para desafiar aún más al algoritmo.

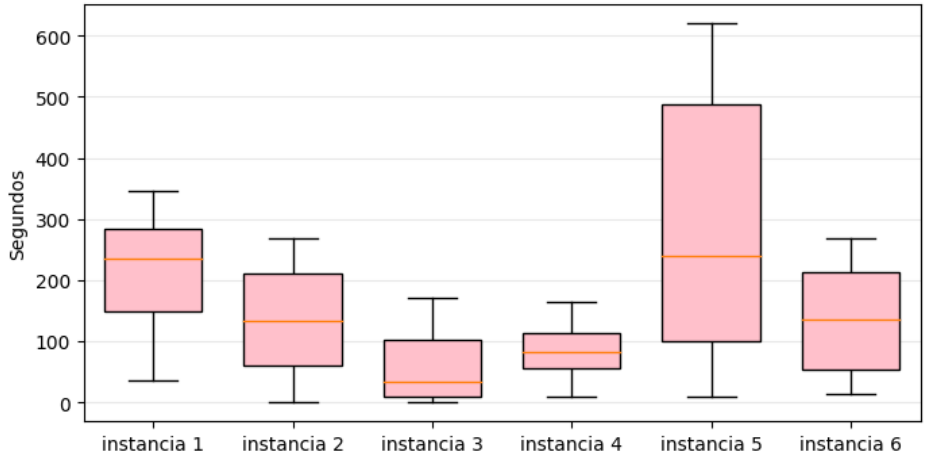


Figura 4.14: Tiempos de ejecución de cada instancia pequeña.

En la figura 4.14 se presenta una comparación entre los tiempos de ejecución de cada instancia pequeña. En ella se observa que la quinta instancia es la que presenta mayor dispersión de los datos. Ésta es la instancia que presenta mayor holgura por lo que la fase de Romper Demandas debe realizar más iteraciones para dejar de satisfacer las demandas y luego la fase Agregar Nodos tiene mucho espacio disponible y demora más en insertar los nodos necesarios para llenar todas las rutas.

Finalmente, es importante notar que el tiempo de ejecución del algoritmo propuesto para las instancias pequeñas en general es de menos de 5 minutos por el contrario, el CPLEX trabajó durante 30 minutos. Aún así, los resultados encontrados por el algoritmo propuesto son muy similares a los obtenidos por el CPLEX.

Como conclusión se puede decir que a mayor holgura peor son los resultados obtenidos por el algoritmo. Esto quiere decir que cuando el algoritmo ya logró suplir las demandas y es libre de sobre abastecer a la planta, comienza a tener un mal desempeño. Por el contrario, cuando la cantidad total demandada de leche es muy cercana a la capacidad máxima de transporte, los resultados obtenidos son de buena calidad.

4.2.4. Evaluación con instancias reales

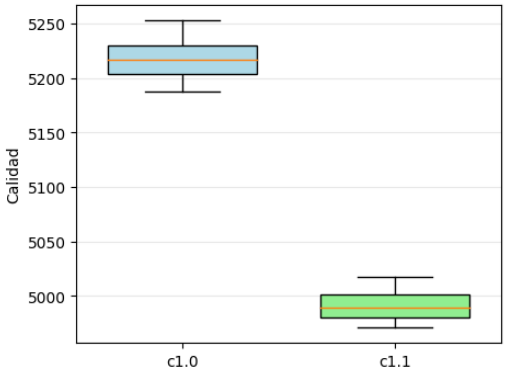
En esta sección se presentan los resultados obtenidos al evaluar el algoritmo en el conjunto de instancias reales. Primero se comparan las soluciones obtenidas para conjunto de nodos considerando las versiones de instancias con y sin demandas de leche. A continuación, se comparan los tiempos de ejecución entre dichas instancias de pruebas.

Para las instancias reales fue necesario adaptar los valores de algunos parámetros que se encontraban estrechamente relacionados con el número de nodos existentes. Los parámetros β, γ y ε fueron ponderados de acuerdo a la cantidad de nodos de cada instancia. Dichos valores se obtuvieron a partir de los valores sintonizados para las instancias pequeñas multiplicados por el número de nodos que posee cada instancia real y dividiendo por 40, que corresponde al total de nodos de las instancias pequeñas como se aprecia en la ecuación 4.1.

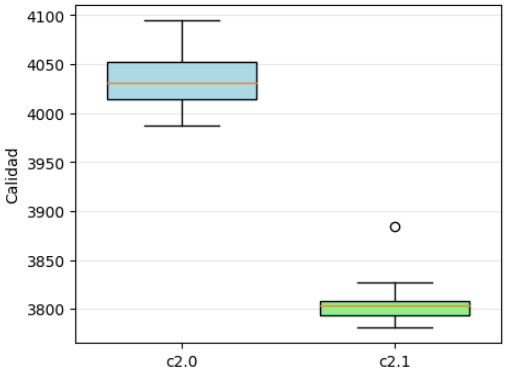
$$param' = \frac{param \times \#nodosInstanciaReal}{40} \quad \forall param \in [\beta = 15, \gamma = 3, \varepsilon = 10] \tag{4.1}$$

Los parámetros τ y PD se mantuvieron en 500 y 0,7 respectivamente pues no poseen una relación tan estrecha con la cantidad de nodos de la instancia.

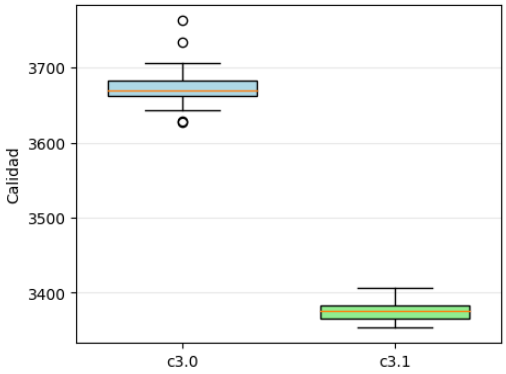
En la figura 4.15 se presentan seis gráficos, uno para cada conjunto de nodos, comparando la instancia que no posee demandas con la que si las posee. La descripción de cada instancia se encuentra en la tabla 4.4 de la sección 4.1.2. Primero se comentan las instancias generadas por el algoritmo k-means y luego se habla de las instancias mayores con 500 nodos.



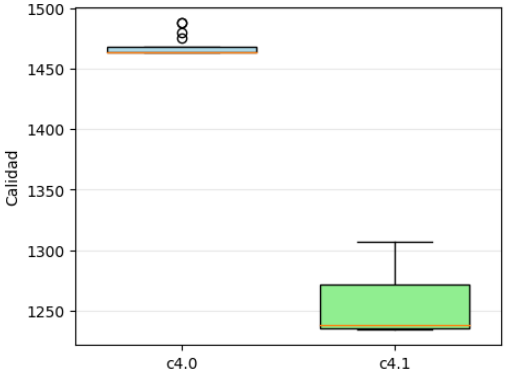
Instancias c1.0 y c1.1



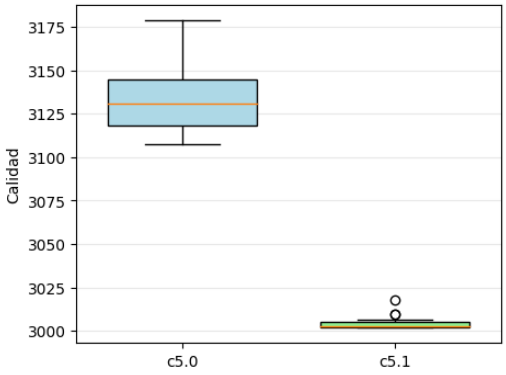
Instancias c2.0 y c2.1



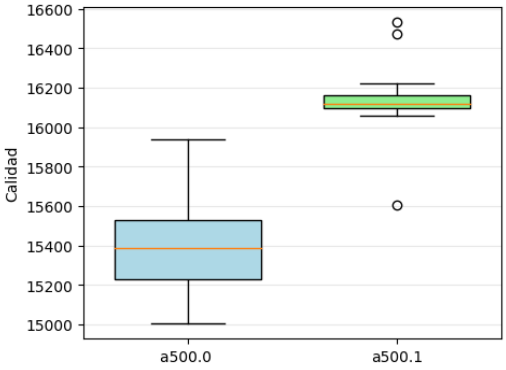
Instancias c3.0 y c3.1



Instancias c4.0 y c4.1



Instancias c5.0 y c5.1



Instancias A500.0 y A500.1

Figura 4.15: Resultados obtenidos en las instancias con demandas contra las instancias sin demandas.

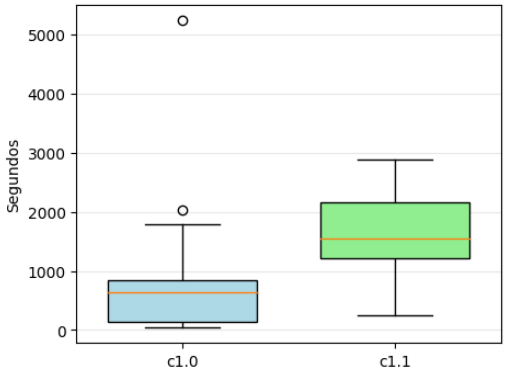
En todas las instancias generadas a partir de los conjuntos de nodos entregados por el algoritmo k-means, se obtiene un desempeño notoriamente mejor cuando las demandas son

cero. Los resultados de estas instancias también poseen una distribución de datos similar, encontrándose en el orden de las $60u.d.$. La instancia $c4{,}1$ es la que presenta la distribución más dispersa dentro de las instancias que poseen demandas. Además, posee un sesgo positivo muy marcado, donde el 50 % de los resultados obtuvo casi el mismo valor. Es precisamente esa instancia la que posee la mayor holgura, teniendo casi un camión entero para sobre abastecer a la planta y por ende, un camión que puede moverse libremente por todos los nodos explorando el espacio de búsqueda.

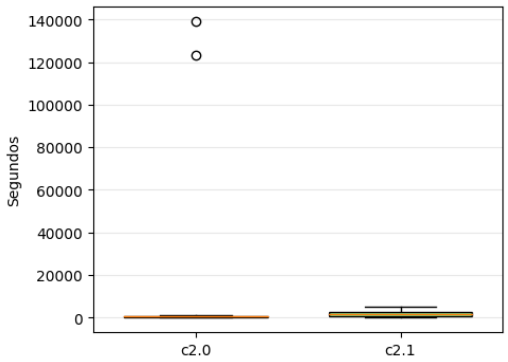
Contrario a los resultados obtenidos por las instancias generadas a partir de subconjuntos de nodos, la instancia mayor que posee 500 nodos obtuvo mejores resultados cuando la planta demanda leche. La mediana de dicha instancia se encuentra $700u.d.$ por sobre la mediana de la instancia sin demandas. Esto se puede producir porque al crear las demandas ficticias, dado que hay una flota muy grande, se pide mucha leche de tipo C. Esto puede generar que se visiten nodos que producen buenas calidades de leche con camiones que llevan la peor calidad, bajando el beneficio obtenido por estos litros. Además, esto deja fuera de las opciones, en las dos fases de diversificación del post-procesamiento, los nodos ya visitados por el camión de tipo C.

No hace sentido comparar numéricamente los valores obtenidos entre los conjuntos pues cada uno está conformado por un distinto número de nodos que poseen diferentes producciones.

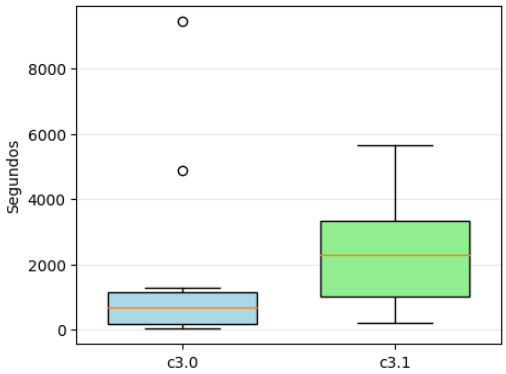
En la figura 4.16 se encuentran los gráficos comparativos de los tiempos de ejecución de cada instancia agrupadas por el conjunto de nodos al que pertenecen. Respecto a estos datos, mirando las medianas de cada boxplot, se puede decir que el algoritmo tarda más tiempo en realizar las 100000 iteraciones cuando existe una demanda realizada por la planta. Además, son esos resultados los que además poseen mayor dispersión de los tiempos. Esto se puede deber a que, por los porcentajes de leche demandados, la fase de Agregar Nodos tarde mucho en lograr volver a una solución factible. Por otro lado, cuando las instancias no demandan leche y se crean demandas ficticias, el tiempo es considerablemente menor obteniendo resultados con hasta $1000seg$ de diferencia. En estos resultados es donde existen outlayers, generados probablemente por aquellas iteraciones donde se genero una configuración de demandas ficticias que era muy complicada de suplir.



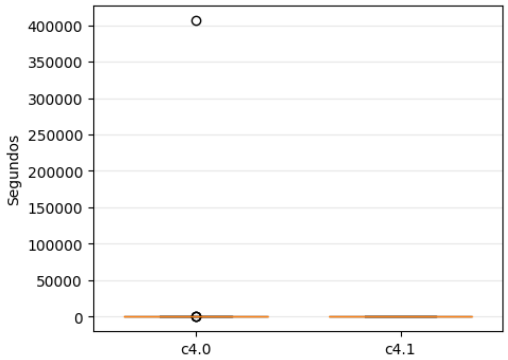
Instancias c1.0 y c1.1



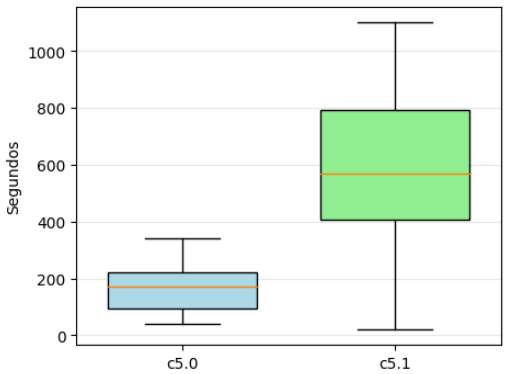
Instancias c2.0 y c2.1



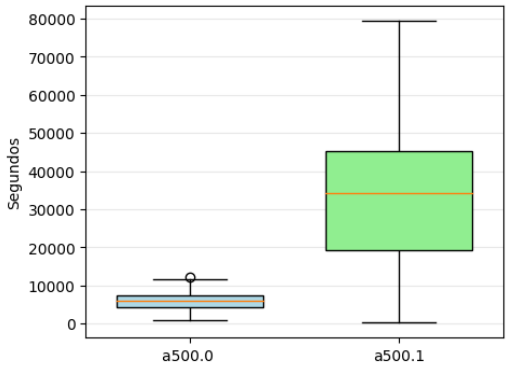
Instancias c3.0 y c3.1



Instancias c4.0 y c4.1



Instancias c5.0 y c5.1



Instancias c500.0 y c500.1

Figura 4.16: Tiempos de ejecución obtenidos en las instancias con demandas versus sin demandas.

Las ocasiones en que las instancias obtuvieron resultados a los pocos segundos de ejecución, se deben a que el algoritmo solo logró construir para algunas iteraciones o que en pocas

ocasiones la fase de Agregar Nodo logró volver a una solución factible, ahorrándose las iteraciones que se realizan cuando si logran su objetivo.

La tabla 4.8 muestra un resumen con el mejor beneficio obtenido por cada instancia real, el tiempo de ejecución en segundos para obtener dicho resultado y los tiempos medios de ejecución en segundos. Como podemos ver, aún para las instancias reales se tienen tiempos de ejecución cercanos a los 38min y la instancia mayor obtiene resultados aproximadamente a las 9hrs y 30min. Estas instancias no podrían obtener resultados en un tiempo razonable si se utiliza el solver CPLEX.

Instancia	**# Nodos**	**Beneficio**	**Tiempo al mejor**	**Tiempo medio**
C1.0	128	5252.80	1783	636
C1.1	128	5018.00	1127	1547
C2.0	143	4094.60	1213	503.5
C2.1	143	3884.40	3396	1850
C3.0	142	3762.70	188	679
C3.1	142	3496.60	2178	2282.5
C4.0	31	1487.90	67	44
C4.1	31	1307.00	152	208.5
C5.0	56	3178.60	101	172.5
C5.1	56	3017.70	549	568
A500.0	500	15937.97	9572	5993
A500.1	500	16531.20	22769	34178

Tabla 4.8: Resultados obtenidos y tiempos del conjunto de instancias grandes

Los beneficios finales obtenidos por cada instancia se ven relacionados con el número de nodos que posee cada una, siendo las más grandes las que obtienen los mejores resultados. La excepción ocurre en el primer conjunto de 128 nodos, donde éste obtiene mejores resultados que los conjuntos $c2$ y $c3$ que poseen sobre 140 nodos. Si miramos la figura 4.3 de la sección 4.1.2, para el primer conjunto de nodos, las medianas de las producciones se encuentran sobre los 2500L en cambio, los conjuntos $c2$ y $c3$ poseen una media igual o menor a los 2000L. Por lo tanto, a pesar de que la primera instancia posea menos nodos que las otras dos,

los nodos que posee producen en general mayores cantidades de leche obteniendo por esa razón mejores resultados finales.

Conclusiones

En el presente trabajo, se investigaron y presentaron las variaciones más típicas del VRP utilizadas en la industria láctea, las cuales fueron apareciendo a medida que se intenta acercar el problema a la realidad. El objetivo en todos ellos es siempre el mismo y considera encontrar las mejores rutas de la flota de vehículos buscando minimizar los costos o aumentar los beneficios.

La aproximación del VRP presentada en este trabajo se denominó MCPSB, por las siglas del inglés Milk Collection Problem with Selection and Blending. El MCPSB considera la recolección de diferentes tipos de leche permitiendo la mezcla tanto en el camión, como en la planta procesadora. Además, considera seleccionar el conjunto de granjas a visitar, ya que la planta demanda una cuota mínima de cada tipo de leche, siendo posible sobre abastecerla siempre y cuando esto genere un beneficio para el problema.

Posteriormente se realiza una propuesta para resolver el MCPSB basada en la meta-heurística Simulated Annealing. El algoritmo implementado posee una fase de construcción enfocada en la producción de los nodos y su cercanía. En esta fase, además de generar soluciones iniciales factibles y de buena calidad, se crean demandas ficticias que son utilizadas en el caso de no poseer demandas por parte de la planta.

El algoritmo propuesto, también considera una fase de post-procesamiento, donde se implementaron cuatro fases que permiten modificar y mejorar las soluciones iniciales. La primera fase del post-procesamiento, nombrada Romper Demandas, busca generar espacio en las rutas para así poder explorar otros sectores del espacio de búsqueda al insertar diferentes nodos

en la siguiente fase llamada Agregar Nodos. En estos dos movimientos es donde la metaheurística toma acción, ya que consideran la temperatura en la que se encuentra el algoritmo y en base a eso aceptan o no, seleccionar nodos que empeoren la solución. Es importante notar que la fase de Romper Demandas genera soluciones infactibles, pues precisamente busca dejar de cumplir las demandas de la planta. Por su parte, la segunda fase, Agregar Nodos, se encarga de volver factible la solución, agregando nodos hasta que no quede espacio en los camiones. En caso de no lograr cumplir las demandas, se vuelve a la solución anterior para comenzar a iterar nuevamente.

Adicionalmente, el algoritmo implementado posee dos fases de intensificación, donde se busca mejorar la solución actual. Estas fases corresponden a Reordenar Nodos y Eliminar Nodos. La primera cambia el orden de visita de los nodos buscando minimizar los recorridos y la segunda elimina los nodos que no generan un beneficio para la solución, siempre manteniendo un estado de factibilidad.

A continuación, se realizó un proceso de sintonización de los parámetros del algoritmo propuesto utilizando ParamILS. Además, se realizaron pruebas por parámetro para analizar el desempeño del algoritmo utilizando el valor sintonizado contra los valores límite con los cuales el parámetro dejaba de hacer efecto o afectaba significativamente las soluciones. A partir del análisis de componentes, se pudo justificar la existencia de cada uno de los parámetros y entender el por qué de su valor sintonizado.

Luego, se realizaron distintos experimentos utilizando dos conjuntos de instancias denominados *instancias de prueba*, formado seis instancias que utilizan un mismo conjunto de 40 nodos, e *instancias reales* con 12 instancias que poseen un universo de hasta 500 nodos.

Respecto a los resultados obtenidos, se observó que el algoritmo propuesto logró tener un muy buen desempeño en las instancias pequeñas, donde fue posible comparar los resultados obtenidos con los generados a partir del modelo de programación entera utilizando el solver CPLEX. En cuatro de las seis instancias de 40 nodos se obtuvo un GAP menor al 5 % obteniendo incluso el mejor resultado para una de ellas. El peor resultado obtuvo un GAP del 19 %. Además, el tiempo de ejecución es menor a los 5*min*, en comparación a los 30*min* que se dejó trabajar al solver.

Las demandas de las instancias reales se crearon a partir de las demandas que tienen las dos instancias de prueba que obtuvieron el mayor GAP. Estos valores consideran no demandar leche y demandar 123 % de la leche de calidad B.

En las cinco instancias que consideran subconjuntos de los 500 nodos, cuando no se poseen demandas se obtuvieron mejores resultados que al sí tenerlas. Por el contrario, el conjunto de 500 nodos obtuvo mucho mejores resultados cuando sí se aplicaba una demanda inicial. Probablemente, esto se genere porque el tipo de leche que genera cada nodo no es el mismo en los subconjuntos que en el conjunto total.

Como trabajo a futuro, se propone modificar el algoritmo para que no se realice mezcla en la planta cuando no existen demandas mínimas. De esta manera, el algoritmo obtiene el beneficio por el tipo de leche que llega en el camión, sin tener que bajar dicha calidad para satisfacer unas demandas que son ficticias. De esta manera, se podrían mejorar los resultados obtenidos por las instancias que no poseen demandas, como es el caso de la instancia pequeña que obtuvo el mayor GAP.

Finalmente, se propone como trabajo a futuro el utilizar puntos de acopio para las instancias mayores. El algoritmo k-means obtiene conjuntos que poseen una buena distribución entre sus nodos, pero que se encuentran a diferentes distancias de la planta. Por esta razón, considerar los centros de los grupos generados por k-means como puntos de acopio podría generar resultados de mejor calidad para instancias grandes.

Anexos

Ejemplo

Para el ejemplo utilizado a lo largo de este documento se utilizaron los siguientes datos:

Leche tipo A				Leche tipo B		Leche tipo C	
nodo	prod.	nodo	prod.	nodo	prod.	nodo	prod.
n_1	10	n_5	15	n_8	25	n_{10}	15
n_2	5	n_6	20	n_9	10		
n_3	20	n_7	10				
n_4	5						

camiones	$k_1 = 45L$	$k_2 = 45L$	
demandas	$A = 35L$	$B = 35L$	$C = 5L$

Tabla 4.9: Datos del ejemplo utilizado.

Granja	Litros	Calidad
1	6438	A
2	2658	B
3	9315	C
4	6575	A
5	4932	B
6	5479	C
7	3288	A
8	2192	B
9	7945	C
10	4568	A
11	7671	B
12	4932	C
13	4795	A
14	2740	B
15	3562	C
16	4384	A
17	7397	B
18	5479	C
19	2470	A
20	13699	B

Granja	Litros	Calidad
21	4658	C
22	2192	A
23	1472	B
24	481	C
25	2899	A
26	497	B
27	10959	C
28	1260	A
29	2395	B
30	2208	C
31	1025	A
32	4658	B
33	548	C
34	548	A
35	4600	B
36	3960	C
37	791	A
38	2000	B
39	20000	C
40	20000	A

Tabla 4.10: Calidad y cantidad de leche producida por granja.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
0	.	134	132	134	133	126	127	108	112	121	109	94	90	139	141	142	121	102	108	113	74	77	37	31	28	26	16	21	11	1	14	18	23	34	42	35	84	86	89	248	305
1	134	.	8	37	36	29	26	31	35	35	31	61	65	85	86	87	97	85	90	95	121	124	167	161	158	156	146	151	141	135	130	126	123	136	144	137	186	188	191	295	352
2	132	8	.	35	34	27	24	29	33	33	29	59	63	83	84	85	95	83	88	93	119	122	165	159	156	154	144	149	139	133	128	24	121	134	142	135	184	186	189	293	350
3	134	37	35	.	5	18	17	28	24	28	31	50	54	99	103	104	104	95	90	93	121	124	167	161	158	156	146	151	141	135	130	126	35	136	144	137	186	188	191	295	352
4	133	36	34	5	.	17	16	27	23	27	30	49	53	98	102	103	103	94	89	92	120	123	166	160	157	155	145	150	140	134	129	125	124	135	143	136	185	187	190	294	351
5	126	29	27	18	17	.	11	18	16	20	23	42	46	91	95	96	96	87	82	85	113	116	159	153	150	148	138	143	133	127	122	118	117	128	143	129	178	180	183	287	344
6	127	26	24	17	16	11	.	21	17	21	16	43	47	92	96	97	97	88	83	86	114	117	160	154	151	149	139	144	134	128	123	119	118	129	144	130	179	181	184	288	345
7	108	31	29	28	27	18	21	.	6	15	5	42	46	84	88	89	78	59	64	67	95	98	141	135	132	130	120	125	115	109	104	100	99	110	125	111	160	162	165	269	326
8	112	35	33	24	23	16	17	6	.	11	9	38	42	88	92	93	82	63	68	71	99	102	145	139	136	134	124	129	119	113	108	104	103	114	129	115	164	166	169	273	330
9	121	35	33	28	27	20	21	15	11	.	18	29	33	97	101	102	91	72	77	80	108	111	154	148	145	143	133	138	128	122	117	113	112	123	138	124	173	175	178	282	339
10	109	31	29	31	30	23	16	5	9	18	.	45	49	85	89	90	79	60	65	68	96	99	142	136	133	131	121	126	116	110	105	101	100	111	126	112	161	163	166	270	327
11	94	61	59	50	49	42	43	42	38	29	45	.	14	119	123	124	118	99	102	105	113	116	127	121	118	116	106	111	101	95	90	86	85	96	104	97	146	148	151	255	312
12	90	65	63	54	53	46	47	46	42	33	49	14	.	123	127	128	122	103	106	109	109	112	123	117	114	112	102	107	97	91	86	82	81	92	100	93	142	144	147	251	308
13	139	85	83	99	98	91	92	84	88	97	85	119	123	.	8	13	23	44	163	166	126	129	172	166	163	161	151	156	146	140	135	131	130	141	149	142	191	193	196	300	357
14	141	86	84	103	102	95	96	88	92	101	89	123	127	8	.	15	25	46	165	168	128	131	174	168	165	163	153	158	148	142	137	133	132	143	151	144	193	195	198	302	359
15	142	87	85	104	103	96	97	89	93	102	90	124	128	13	15	.	26	47	166	169	129	132	175	169	166	164	154	159	149	143	138	134	133	144	152	145	194	196	199	303	360
16	121	97	95	104	103	96	97	78	82	91	79	118	122	23	25	26	.	27	64	67	108	111	154	148	145	143	133	138	128	122	117	113	112	123	131	124	173	175	178	282	339
17	102	85	83	95	94	87	88	59	63	72	60	99	103	44	46	47	27	.	45	48	89	92	135	129	126	124	114	119	109	103	98	94	93	104	112	105	154	156	159	263	320
18	108	90	88	90	89	82	83	64	68	77	65	102	106	163	165	166	64	45	.	7	95	98	141	135	132	130	120	125	115	109	104	100	99	110	118	111	160	162	165	269	326
19	113	95	93	93	92	85	86	67	71	80	68	105	109	166	168	169	67	48	7	.	100	103	146	140	137	135	125	130	120	114	109	105	104	115	118	116	165	167	170	274	331
20	74	121	119	121	120	113	114	95	99	108	96	113	109	126	128	129	108	89	95	100	.	13	59	53	50	56	62	73	67	75	84	88	93	104	112	106	155	157	160	264	321
21	77	124	122	124	123	116	117	98	102	111	99	116	112	129	131	132	111	92	98	103	13	.	62	56	53	59	65	76	70	78	87	91	96	107	115	109	158	160	163	267	324
22	37	167	165	167	166	159	160	141	145	154	142	127	123	172	174	175	154	135	141	146	59	62	.	8	13	19	25	36	30	38	47	51	56	67	75	68	117	119	122	226	283
23	31	161	159	161	160	153	154	135	139	148	136	121	117	166	168	169	148	129	135	140	53	56	8	.	7	13	19	30	24	32	41	45	50	61	69	62	111	113	116	275	332
24	28	158	156	158	157	150	151	132	136	145	133	118	114	163	165	166	145	126	132	137	50	53	13	7	.	10	16	27	21	29	38	42	47	58	66	59	108	110	113	272	329
25	26	156	154	156	155	148	149	130	134	143	131	116	112	161	163	164	143	124	130	135	56	59	19	13	10	.	14	25	19	27	36	40	45	56	64	57	106	108	111	270	327
26	16	146	144	146	145	138	139	120	124	133	121	106	102	151	153	154	133	114	120	125	62	65	25	19	16	14	.	15	9	17	26	30	35	46	54	47	96	98	101	260	317
27	21	151	149	151	150	143	144	125	129	138	126	111	107	156	158	159	138	119	125	130	73	76	36	30	27	25	15	.	14	22	31	35	40	51	59	52	101	103	106	265	322
28	11	141	139	141	140	133	134	115	119	128	116	101	97	146	148	149	128	109	115	120	67	70	30	24	21	19	9	14	.	12	21	25	30	41	49	42	91	93	96	255	312
29	1	135	133	135	134	127	128	109	113	122	110	95	91	140	142	143	122	103	109	114	75	78	38	32	29	27	17	22	12	.	15	19	24	35	43	36	85	87	90	249	306
30	14	130	128	130	129	122	123	104	108	117	105	90	86	135	137	138	117	98	104	109	84	87	47	41	38	36	26	31	21	15	.	14	19	30	38	31	85	87	90	249	306
31	18	126	24	126	125	118	119	100	104	113	101	86	82	131	133	134	113	94	100	105	88	91	51	45	42	40	30	35	25	19	14	.	15	26	34	27	81	83	86	245	302
32	23	123	121	35	124	117	118	99	103	112	100	85	81	130	132	133	112	93	99	104	93	96	56	50	47	45	35	40	30	24	19	15	.	21	29	22	76	78	81	240	297
33	34	136	134	136	135	128	129	110	114	123	111	96	92	141	143	144	123	104	110	115	104	107	67	61	58	56	46	51	41	35	30	26	21	.	12	19	69	71	74	178	235
34	42	144	142	144	143	143	144	125	129	138	126	104	100	149	151	152	131	112	118	118	112	115	75	69	66	64	54	59	49	43	38	34	29	12	.	27	77	79	82	186	243
35	35	137	135	137	136	129	130	111	115	124	112	97	93	142	144	145	124	105	111	116	106	109	68	62	59	57	47	52	42	36	31	27	22	19	27	.	51	53	56	160	217
36	84	186	184	186	185	178	179	160	164	173	161	146	142	191	193	194	173	154	160	165	155	158	117	111	108	106	96	101	91	85	85	81	76	69	77	51	.	6	15	127	184
37	86	188	186	188	187	180	181	162	166	175	163	148	144	193	195	196	175	156	162	167	157	160	119	113	110	108	98	103	93	87	87	83	78	71	79	53	6	.	17	129	186
38	89	191	189	191	190	183	184	165	169	178	166	151	147	196	198	199	178	159	165	170	160	163	122	116	113	111	101	106	96	90	90	86	81	74	82	56	15	17	.	132	189
39	248	295	293	295	294	287	288	269	273	282	270	255	251	300	302	303	282	263	269	274	264	267	226	275	272	270	260	265	255	249	249	245	240	178	186	160	127	129	132	.	143
40	305	352	350	352	351	344	345	326	330	339	327	312	308	357	359	360	339	320	326	331	321	324	283	332	329	327	317	322	312	306	306	302	297	235	243	217	184	186	189	143	.

Tabla 4.11: Matriz de distancias de las instancias utilizadas.

Bibliografía

[1] *Gateway to dairy production and products*. http://www.fao.org/dairy-production-products/processing/collection-and-transport/en/. Accessed: 2018-01-22.

[2] Aarts, Emile HL, Jan HM Korst y Peter JM van Laarhoven: *A quantitative analysis of the simulated annealing algorithm: A case study for the traveling salesman problem*. Journal of Statistical Physics, 50(1-2):187–206, 1988.

[3] Alfa, Attahiru Sule, Sundresh S Heragu y Mingyuan Chen: *A 3-opt based simulated annealing algorithm for vehicle routing problems*. Computers & Industrial Engineering, 21(1-4):635–639, 1991.

[4] Bing, Xiaoyun, Marlies de Keizer, Jacqueline M Bloemhof-Ruwaard y Jack GAJ van der Vorst: *Vehicle routing for the eco-efficient collection of household plastic waste*. Waste management, 34(4):719–729, 2014.

[5] Bodin, Lawrence: *Routing and scheduling of vehicles and crews, the state of the art*. Comput. Oper. Res., 10(2):63–211, 1983.

[6] Caramia, Massimiliano y Francesca Guerriero: *A milk collection problem with incompatibility constraints*. Interfaces, 40(2):130–143, 2010.

[7] Chite, Ralph: *Milk Standards: Grade A vs. Grade B*. Congressional Research Service, Library of Congress, 1991.

[8] Christofides, Nicos y Samuel Eilon: *An algorithm for the vehicle-dispatching problem*. Journal of the Operational Research Society, 20(3):309–318, 1969.

[9] Dantzig, George B y John H Ramser: *The truck dispatching problem*. Management science, 6(1):80–91, 1959.

[10] Dantzig, Tobias: *Number the Language of Science (1930)*, 2003.

[11] Dayarian, Iman, Teodor Gabriel Crainic, Michel Gendreau y Walter Rei: *A branch-and-price approach for a multi-period vehicle routing problem*. Computers & Operations Research, 55:167–184, 2015.

[12] Dayarian, Iman, Teodor Gabriel Crainic, Michel Gendreau y Walter Rei: *A column generation approach for a multi-attribute vehicle routing problem*. European Journal of Operational Research, 241(3):888–906, 2015.

[13] Dayarian, Iman, Teodor Gabriel Crainic, Michel Gendreau y Walter Rei: *An adaptive large-neighborhood search heuristic for a multi-period vehicle routing problem*. Transportation Research Part E: Logistics and Transportation Review, 95:95–123, 2016.

[14] Dooley, AE, WJ Parker y HT 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.

[15] El Fallahi, Abdellah, Christian Prins y Roberto Wolfler Calvo: *A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem*. Computers & Operations Research, 35(5):1725–1741, 2008.

[16] Gendreau, Michel, Alain Hertz y Gilbert Laporte: *New insertion and postoptimization procedures for the traveling salesman problem*. Operations Research, 40(6):1086–1094, 1992.

[17] Gutiérrez, Miguel Ángel, Sergio de los Cobos y Blanca Pérez: *Op timización con recocido simulado para el problema de conjunto inde pendiente*. Universidad Autónoma Metropolitana, México, disponible en: http://www. azc. uam. mx/publicaciones/enlinea2/3 2rec. htm,[fecha de consulta: 12 de enero de 2007], 1998.

[18] Hutter, Frank, Holger H Hoos, Kevin Leyton-Brown y Thomas Stützle: *ParamILS: an automatic algorithm configuration framework*. Journal of Artificial Intelligence Research, 36:267–306, 2009.

[19] INIFAP, instituto nacional de investigaciones forestales, agrícolas y pecuarias: *Mejora continua de la calidad higiénico sanitaria de la leche de vaca*, Mayo 2011. `http://utep.inifap.gob.mx/pdf_s/MANUAL%20LECHE.pdf`.

[20] Junqueira, Leonardo, José F Oliveira, Maria Antónia Carravilla y Reinaldo Morabito: *An optimization model for the vehicle routing problem with practical three-dimensional loading constraints*. International Transactions in Operational Research, 20(5):645–666, 2013.

[21] Kirkpatrick, Scott, C Daniel Gelatt y Mario P Vecchi: *Optimization by simulated annealing*. science, 220(4598):671–680, 1983.

[22] Lahrichi, Nadia, Teodor Gabriel Crainic, Michel Gendreau, Walter Rei y Louis Martin Rousseau: *Strategic analysis of the dairy transportation problem*. Journal of the Operational Research Society, 66(1):44–56, 2015.

[23] Laporte, Gilbert, Michel Gendreau, J Y Potvin y Frédéric Semet: *Classical and modern heuristics for the vehicle routing problem*. International transactions in operational research, 7(4-5):285–300, 2000.

[24] Liu, Shuguang, Lei Lei y Sunju Park: *On the multi-product packing-delivery problem with a fixed route*. Transportation Research Part E: Logistics and Transportation Review, 44(3):350–360, 2008.

[25] Lloyd, SP: *Least square quantization in PCM. Bell Telephone Laboratories Paper. Published in journal much later: Lloyd, SP: Least squares quantization in PCM*. IEEE Trans. Inform. Theor.(1957/1982), 18, 1957.

[26] Metropolis, Nicholas, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller y Edward Teller: *Equation of state calculations by fast computing machines*. The journal of chemical physics, 21(6):1087–1092, 1953.

[27] Panamá, La estrella de: *Lecheros piden 0.15 por litro natural*, Marzo 2015. `http://laestrella.com.pa/economia/lecheros-piden-015-litro-natural/23853289`.

[28] Paredes-Belmar, German, Armin Lüer-Villagra, Vladimir Marianov, Cristian E Cortes y Andres Bronfman: *The milk collection problem with blending and collection points*. Computers and Electronics in Agriculture, 134:109–123, 2017.

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

[30] Raff, Samuel: *Routing and scheduling of vehicles and crews: The state of the art*. Computers & Operations Research, 10(2):63–211, 1983.

[31] Sankaran, Jayaram K y Rahul R Ubgade: *Routing tankers for dairy milk pickup*. Interfaces, 24(5):59–66, 1994.

[32] Sethanan, Kanchana y Rapeepan Pitakaso: *Differential evolution algorithms for scheduling raw milk transportation*. Computers and Electronics in Agriculture, 121:245–259, 2016.

[33] Taillard, Éric: *Parallel iterative search methods for vehicle routing problems*. Networks, 23(8):661–673, 1993.