

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



“HERRAMIENTA DE APOYO A LA TOMA DE DECISIONES
PARA EL PROBLEMA DE RECOLECCIÓN DE LA LECHE
CON MEZCLA”

ANDRÉS FELIPE NAVARRO GALLEGUILLOS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: María Cristina Riff Rojas
Profesor Correferente: Elizabeth Montero Ureta

Marzo - 2022

DEDICATORIA

Sin mi madre no habría podido llegar a donde estoy actualmente. Este trabajo está a ella, Janet Galleguillos, gracias por toda tu preocupación, valores y amor que he recibido desde que tengo recuerdos.

AGRADECIMIENTOS

En primera instancia quiero agradecer la profesora María Cristina Riff, quien me dio la posibilidad de desarrollar el tema de esta memoria, brindándome sus conocimientos y además orientándome en mi futuro profesional. Agradezco que haya escuchado todos mis problemas (de diversa índole) y me haya ayudado en todos estos.

Quiero agradecer a mi madre, Janet Galleguillos, por haberme criado y hacer todo lo posible para que pudiese llegar a estudiar en una universidad. Siempre has sido alguien que ha estado ahí para mí, sin importar la situación y ayudándome de cualquier forma que se le ocurriese

Mis amigos y compañeros que conocí en la universidad fueron una parte importante de mi vida universitaria, estudiando, pasándola bien y preocupándonos los unos de los otros. Si bien no podría nombrar a todos quiero recordar especialmente a Sonny Muñoz, Sebastian Muñoz, Javier Moreno, Matías Santander y Javier Castro. No puedo olvidar a Kevin Lagos, con quien decidimos realizar un magíster, por lo que me seguiré topando con él en la universidad un tiempo más.

Agradezco también a mis amistades del liceo que aún conservo, David Morales, Camilo Vásquez y Héctor Carvajal y a mis amistades que conocí por amigos de la universidad, Matías Figueroa, Nicolás Carrillo, Sebastián Vera y Marcelo Varas.

No puedo olvidar mencionar a mi polola, Bárbara Uribe, quien me ha acompañado, primero como amigos, desde el primer año de la universidad. El apoyo, comprensión, amor que me has dado y los recuerdos que hemos formado son increíbles.

Por último, quiero agradecer al streamer que me acompañó y entretuvo en todas mis horas de trabajo, Alejandro Fernández o Alexelcapo, muchas gracias por el contenido creado.

RESUMEN

Resumen—

El problema de recolección de leche con mezcla es un problema de optimización combinatoria. Este debe encontrar las rutas óptimas para camiones que recogen leche de diferentes granjas para llevarla a una planta procesadora, optimizando los beneficios. Las granjas producen cierto tipo de leche que determinando su calidad. Al ser recogida, es mezclada con la leche transportada, resultando en la de menor calidad. Esto puede reducir los beneficios, pero permite generar rutas menos costosas. Se propone diseñar una herramienta para apoyar la toma de decisiones, basada en una técnica de búsqueda local para obtener las rutas que optimicen las ganancias.

Palabras Clave—

Recolección de leche; Mezcla de leche; Algoritmo heurístico; Búsqueda local

ABSTRACT

Abstract— The milk collection problem with blending is a combinatorial optimization problem. It must find the optimal routes for trucks that collect milk from different farms to take it to a processing plant, optimizing profits. Farms produce a certain type of milk that determines its quality. When collected, it is blended with the transported milk, resulting in the lower quality milk. This may result in lower profits, but allows less expensive routes to be generated. It is proposed to design a decision making support tool, based on a local search technique to obtain the routes that optimize profits.

Keywords—

Milk Collection; Milk Blending; Heuristic Algorithm; Local search

GLOSARIO

ADTP: Annual Dairy Transportation Problem
ALNS: Adaptative Large-Neighborhood Search
DSS: Decision Support System
DTP: Dairy Transportation Problem
GUTS: Generalized Unified Tabu Search
HFFVRP: Heterogeneous Fixed Fleet Vehicle Routing Problem
ILS: Iterated Local Search
IP: Integer Programming
MAVRP: Multi-attribute Vehicle Routing Problem
MBCP: Milk Collection Problem with Blending and Collection Points
MCP: Milk Collection Problem
MCVRP: Multi-compartment Vehicle Routing Problem
MCwB: Milk Collection Problem with Blending
MIP: Mixed Integer Programming
MPVRP: Multi-Product Vehicle Routing Problem
TSP: Traveling Salesman Problem
TSPTW: Traveling Salesman Problem with Time Windows
TTRP: Truck and Trailer Routing Problem
UM: Unidades monetarias
UTFSM: Universidad Técnica Federico Santa María.
VRP: Vehicle Routing Problem

ÍNDICE DE CONTENIDOS

RESUMEN	IV
ABSTRACT	IV
GLOSARIO	V
ÍNDICE DE FIGURAS	VIII
ÍNDICE DE TABLAS	VIII
INTRODUCCIÓN	1
CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA	3
1.1 Modelo matemático	6
1.1.1 Parámetros	7
1.1.2 Variables	8
1.1.3 Restricciones	8
1.1.4 Naturaleza de las variables	10
1.1.5 Función objetivo	10
1.2 Objetivos	11
1.2.1 Objetivo General	11
1.2.2 Objetivos específicos	11
CAPÍTULO 2: ESTADO DEL ARTE	12
2.1 VRP y variantes	12
2.1.1 Vehicle Routing Problem	12
2.1.2 Multi-Product Vehicle Routing Problem	13
2.1.3 Multi-compartment Vehicle Routing Problem	13
2.2 Problemas con un tipo de leche	14
2.2.1 Primeros acercamientos	14
2.2.2 Two-Period Travelling Salesman Problem	14
2.2.3 Heterogeneous Fixed Fleet Vehicle Routing Problem	14
2.2.4 Traveling Salesman Problem with Time Windows	15
2.2.5 Truck and Trailer Routing Problem	15
2.2.6 Scheduling and Sequencing Problem	15
2.2.7 Annual Dairy Transportation Problem	16
2.3 Problemas con más de un tipo de leche, sin mezcla	16
2.3.1 Algoritmos genéticos	16
2.3.2 Truck and Trailer Routing Problem	17
2.3.3 Multi-attribute Vehicle Routing Problem	17
2.3.4 Multi-period Vehicle Routing Problem	18

2.4 Problemas con más de un tipo de leche, con mezcla	18
2.4.1 Milk Collection Problem with Blending	18
2.4.2 Milk Colection Problem with Blending and Collection Points	22
CAPÍTULO 3: PROPUESTA DE SOLUCIÓN	23
3.1 Representación	23
3.2 Función de evaluación	24
3.3 Generación de soluciones iniciales	26
3.4 Algoritmo a implementar	30
3.4.1 Movimiento de reubicación de granja	31
3.4.2 Movimiento de intercambio de camiones	33
3.4.3 Generación y codificación de los movimientos	35
3.4.4 Hill-Climbing	40
CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN	43
4.1 Objetivos	43
4.2 Instancias de prueba	43
4.2.1 Instancias pequeñas del estado del arte	44
4.2.2 Instancias medianas del estado del arte	46
4.2.3 Instancia Real	47
4.3 Entorno de experimentación	49
4.4 Experimentación	49
4.4.1 Instancias pequeñas	50
4.4.2 Instancias medianas	55
4.4.3 Instancia Real	60
4.4.4 Utilidad de movimiento de intercambio de camiones	63
CAPÍTULO 5: CONCLUSIONES	66
REFERENCIAS BIBLIOGRÁFICAS	68

ÍNDICE DE FIGURAS

1	Ejemplo de solución a instancia con 14 granjas y 3 camiones.	5
2	Ejemplo de instancia pequeña con 3 camiones y 21 granjas	24
3	Ejemplo de instancia pequeña con 3 camiones y 21 granjas	30
4	Ejemplo de instancia pequeña con 3 camiones y 21 granjas	37
5	Posiciones posibles enumeradas en una instancia de 21 granjas	37
6	Inserciones posibles en las rutas de una instancia	38
7	Ejemplo de instancia con 5 rutas y camiones	39
8	Rutas con las que intercambiar según el vector de códigos	40
9	Distribución de las granjas en la región geográfica	48
10	Boxplot en base a las calidades obtenidas por los 10 experimentos para cada número de camiones	63
11	Boxplot de comparación de las calidades obtenidas por los 10 experimentos para cada número de camiones, con 1 y 2 movimientos	65

ÍNDICE DE TABLAS

1	Requerimientos de la planta procesadora.	6
2	Instancias obtenidas del trabajo de Reinelt.	20
3	Instancias obtenidas del trabajo de Fisher.	20
4	Instancias obtenidas del trabajo de Taillard.	20
5	Instancias obtenidas del trabajo de Augerat.	21
6	Configuración de soluciones iniciales para instancias del estado del arte	30
7	Instancias pequeñas del estado del arte.	45
8	Instancias medianas del estado del arte.	46

9	Parámetros para la ejecución del algoritmo y valores utilizados para instancias pequeñas.	51
10	Resultados obtenidos con distintos parámetros de penalización para instancias pequeñas.	52
11	Mejores resultados obtenidos para instancias pequeñas.	54
12	Parámetros para la ejecución del algoritmo y valores utilizados para instancias medianas.	56
13	Resultados obtenidos para distintas configuraciones iniciales de instancias medianas.	57
14	Resultados obtenidos para distintas configuraciones iniciales de instancias medianas.	58
15	Resultados obtenidos para distintas configuraciones iniciales de instancias medianas.	59
16	Mejores resultados obtenidos para instancias medianas.	60
17	Parámetros para la ejecución del algoritmo y valores utilizados para instancia real.	61
18	Resultados obtenidos con distintos número de camiones para instancia real. . .	62
19	Mejor resultado obtenido para instancia real.	62
20	Resultados obtenidos con distinto número de camiones para instancia real sin movimiento de intercambio de camiones.	64

INTRODUCCIÓN

Los productos lácteos han estado presentes en la historia de la humanidad desde hace muchísimo tiempo, siendo la leche el principal producto a consumir y materia prima de muchos otros alimentos consumidos diariamente. Actualmente, en la industria láctea, la recolección de la materia prima, la leche, corresponde a uno de los costos más relevantes a considerar. Esto está dado porque la leche es un producto perecedero, que presenta ventanas de tiempo muy acotadas en las que se puede mantener por los productores una vez es extraída, hasta que lleguen los camiones recolectores. Si además se considera que, en muchas ocasiones las granjas productoras pueden ubicarse en sitios de difícil acceso o lejanos a la planta procesadora, se convierte en un desafío de coordinación no menor, que necesita de rutas que consideren a todas las granjas y los tiempos en que la leche es ordeñada.

Sumado a lo anterior, existen distintos tipos de leche que las granjas pueden producir, teniendo que considerar lo que la planta exige de cada tipo existente y los distintos beneficios que estas aportan por su calidad. Para la recolección de estas, se necesitan camiones que contengan varios compartimentos para poder recolectar toda la leche que se produce de los distintos tipos, o bien, considerar un número mayor de estos para recolectar los distintos tipos sin que se mezclen.

Sin embargo, existe una alternativa a esto, poder mezclar distintos tipos de leche en un mismo compartimento o camión (si es que este sólo cuenta con uno) degradando así la totalidad de la leche a la que posea una menor calidad, práctica que es ampliamente utilizada [GLH Incorporated, 2014]. Lo anterior puede reducir el beneficio aportado por la leche transportada, pero abre la posibilidad a generar rutas con un menor coste, pudiendo obtener una planificación para los camiones de tal manera que se aumenten las ganancias totales en comparación a una que no permita la mezcla de leches.

Considerando lo anterior, se plantea construir un algoritmo basado en una técnica de búsqueda local para resolver el problema de recolección de leche con mezcla, en el que se tienen un conjunto de granjas que producen distintos tipos de leche y en distintas cantidades, una flota de camiones con un solo compartimento y la obligación de cumplir con cuotas de leche de cada tipo definidas por la planta procesadora. Siendo permitida la mezcla de leches de distinto tipo, ya sea en un mismo camión, o bien luego de entregarla en la planta procesadora.

Este trabajo se estructura de la siguiente manera: en el primer capítulo se presenta la definición del problema a abordar, el objetivo general y los objetivos específicos. Luego, en el capítulo 2, se realiza una revisión de la literatura relacionada al problema a tratar, considerando variantes de VRP, problemas de recolección de leche con un sólo tipo y con varios tipos, con o sin mezcla. En el capítulo 3 se presenta el algoritmo basado en *Hill-Climbing* desarrollado para resolver el problema, abordando la representación utilizada, la generación de soluciones iniciales, los movimientos empleados y la forma en que se construyen los ve-

cinos. Posteriormente, en el cuarto capítulo se presentan las instancias con las que trabajará el algoritmo y los resultados que se obtienen al realizar distintos tipos de pruebas, comparando lo obtenido con la literatura. Finalmente, en el capítulo 5 se exponen las conclusiones obtenidas del trabajo realizado y el trabajo a futuro que se podría desarrollar en torno a este problema.

CAPÍTULO 1

DEFINICIÓN DEL PROBLEMA

En la industria láctea, la recolección de materias primas es una tarea crucial, ya que las ganancias dependen fundamentalmente del costo que conlleva recolectar la leche de los distintos productores, proceso que se realiza a través de numerosas paradas en distintas granjas productoras de leche. Sin embargo, esta tarea no está exenta de dificultades, ya que muchas veces estos productores se encuentran lejos unos de otros, estando separados por áreas rurales de gran extensión, provocando que el transporte de esta materia sea costoso, a lo que se debe añadir el hecho de que una vez la leche es ordeñada, esta debe ser recolectada dentro de 4 horas [FAO, 2002], lo que puede llegar a presentar desafíos de coordinación con los mismos productores en base, a las rutas de recolección que se pudiesen llegar a establecer.

Por otro lado, se debe considerar que los productores, a menudo, se agrupan en pequeñas cooperativas, que se tratan como una entidad consolidada con un solo comprador, como se aborda en [Prasertsri y Kilmer, 2004], provocando que se tenga que recoger la leche de todos los miembros de la cooperativa, a pesar de que esto pueda no ser rentable debido a la ubicación de algunos. Además, se debe considerar el hecho de que no todas las granjas producen la misma cantidad de leche y que la calidad de esta misma puede variar de una granja a otra, aportando más beneficios a la planta procesadora entre mejor sea su calidad.

Otro factor a considerar es que, debido a que la leche es un perecedero, debe ser conservada en frío, con el objetivo de retardar o detener el crecimiento de bacterias, conservándola antes de que sea tratada para generar diversos productos lácteos. Es por lo anterior que se deben usar camiones con almacenamiento refrigerado para su transporte, cuyo costo es mayor que uno tradicional, además de tener que considerar costos asociados a la desinfección y limpieza de los mismos.

Teniendo en cuenta lo anterior, en 2016 se plantea por primera vez el problema de recolección de leche con mezcla [Paredes-Belmar *et al.*, 2016] por Paredes-Belmar. En este, se considera un conjunto de granjas que produce una determinada cantidad de leche de cierto tipo, pudiendo ser A, B, o C, que está ligada a la calidad de la misma, siendo la de tipo A la mayor calidad y que por ende aportará la mayor cantidad de beneficios por litro a la planta procesadora, mientras que la de tipo C será la de menor calidad. Además, la leche producida por la granja se debe recolectar solo una vez, por lo que el camión encargado de recolectar la leche de una determinada granja debe tener la capacidad suficiente para recolectar la totalidad de esta. Adicional a esto, la planta procesadora exige cuotas mínimas de cada tipo de leche.

Por otro lado, se tiene una flota heterogénea de camiones (es decir, que pueden tener distinta capacidad total), que poseen un sólo compartimento para almacenar toda la leche que recolecten en su ruta, la que debe iniciar en la planta procesadora, luego visitar una cierta cantidad de granjas y recolectar la leche producida por estas, para finalmente regresar

a la planta procesadora. Cada camión posee una capacidad determinada que no puede ser excedida y toda la leche que recolecten será mezclada en su tanque.

La mezcla que se realiza en el camión se plantea de tal manera que, dado que cada granja produce un tipo específico, si la leche recolectada posee una calidad distinta a la almacenada hasta el momento en el camión, la mezcla resultante será leche del tipo de peor calidad. La planta procesadora, de ser necesario, también puede mezclar la leche entregada por todos los camiones, en caso de que sea necesario, para cumplir las cuotas exigidas. Es pertinente aclarar que la mezcla de leches como se plantea en el problema, es una práctica común [GLH Incorporated, 2014] y mientras se clasifique la leche resultante de forma correcta, no se viola ninguna regulación.

Los beneficios aportados a la planta procesadora están determinados por la cantidad y el tipo de leche que es entregado a esta por cada camión, pudiendo realizar una mezcla adicional en la misma planta en caso de que esto sea factible. Mientras que, para los costos asociados, se tiene una matriz de distancia entre todos los nodos, los que corresponderían a las granjas y la planta recolectora, pudiendo a través de esta, determinar finalmente el costo de la ruta realizada por cada camión.

En este problema, el objetivo consiste en maximizar la ganancia de la planta a través de optimizar el método de recolección de leche, considerando que esta se puede mezclar en ruta y tomando en cuenta una flota heterogénea de camiones. Así, la solución de este problema permite determinar las rutas que cada camión debe tomar, lo que corresponde a las granjas que debe visitar y el orden en que las visita.

El componente de mezcla de este problema se hace relevante ya que, si bien se están reduciendo los beneficios al disminuir la calidad de la leche, esto da paso a la posibilidad de generar otras rutas que den como resultado una reducción en los costos de transporte mayor a la pérdida del beneficio, generando finalmente un balance positivo. Se debe considerar también, que alternativas a esta característica, como lo sería usar un camión con varios compartimentos, o simplemente usar camiones distintos para cada calidad, son costosas, más aún si se considera que algunas granjas poseen producciones pequeñas.

A continuación, se presentará un ejemplo de solución a una instancia de este problema, que es presentada en la figura 1. En este se deben considerar los siguientes datos:

- Se tiene una planta procesadora, correspondiente al nodo 0 con color gris.
- Se tienen tres tipos de leche, cada uno con distinta calidad:
 - Leche de tipo A, de la mayor calidad entre las tres, que cuenta con 5 granjas que se identifican con nodos de color azul.
 - Leche de tipo B, de calidad intermedia, que cuenta con 4 granjas que se identifican con nodos de color verde.

- Leche de tipo C, de la peor calidad entre las tres, que cuenta con 4 granjas que se identifican con nodos de color naranja.
- Se tienen tres camiones en total, por lo que necesariamente un camión recolectará un tipo de leche en específico y sus rutas poseen el color de la leche que recolectaron. Cada ruta presenta el costo de traslado en unidades monetarias (UM) de un nodo a otro con un número encima de las flechas entre estos.
- Cada nodo posee un valor fuera de este que corresponde a la producción de su leche en litros.

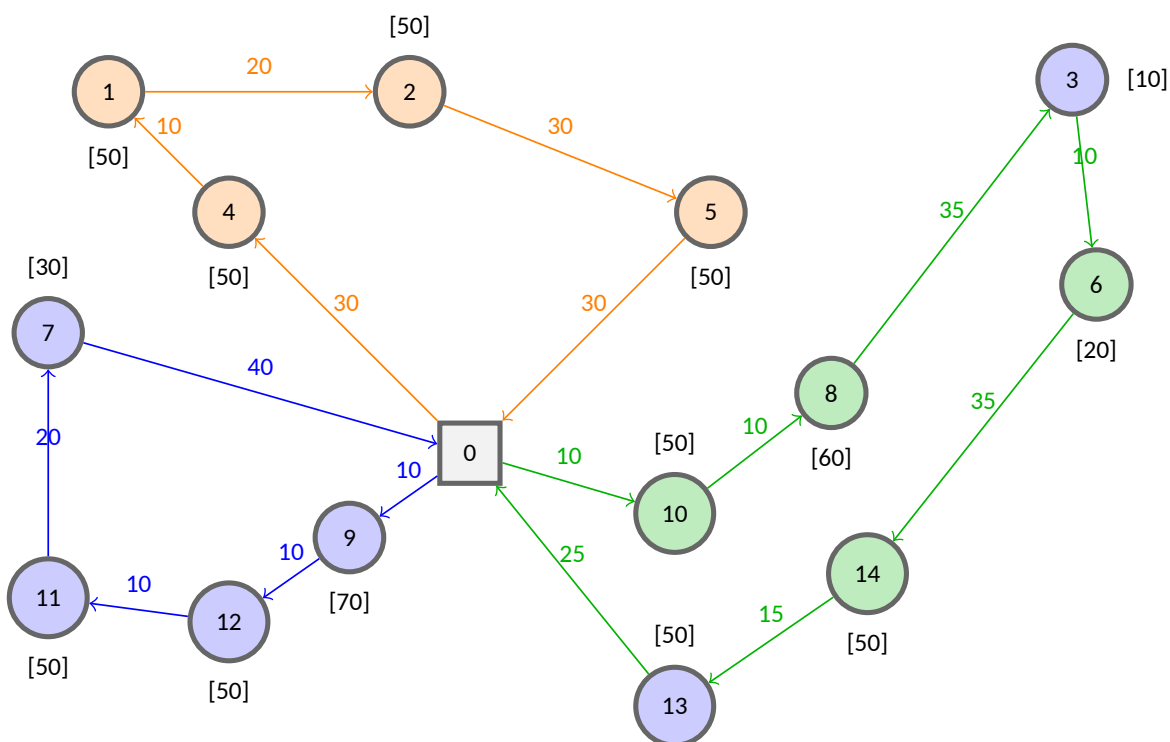


Figura 1: Ejemplo de solución a instancia con 14 granjas y 3 camiones.

Fuente: Elaboración Propia.

Por otro lado, en la tabla 1 se pueden ver los requerimientos de la planta de cada tipo de leche, en donde se presenta el beneficio de estas en UM por cada litro y la cuota mínima que la planta procesadora exige.

Con la información dada se puede calcular los costos y beneficios aportados por las rutas de los tres camiones:

- Ruta A: recolecta 200 litros de leche de tipo A, posee un costo de transporte igual a 90 UM y un ingreso igual a 200 UM.

- Ruta B: recolecta 240 litros de leche de tipo B, posee un costo de transporte igual a 140 UM y un ingreso igual a 168 UM.
- Ruta C: recolecta 200 litros de leche de tipo C, posee un costo de transporte igual a 120 UM y un ingreso igual a 60 UM.

Tabla 1: Requerimientos de la planta procesadora.

Fuente: Elaboración Propia.

Tipo de leche	Aporte monetario	Cuota mínima
A	1 UM	180 litros
B	0.7 UM	200 litros
C	0.3 UM	220 litros

Respecto a esta propuesta de solución se puede destacar que, al estar las granjas 3 y 13 alejadas del resto de granjas que producen leche de tipo A, el camión asociado a este tipo no pasó por estas, ya que podría incurrir en costos significativos, por lo que estas formaron parte de la ruta para la leche de tipo B, ya que el costo sería menor, ocurriendo en la ruta la mezcla de estas y degradando su calidad a leche de tipo B. Por otro lado, se hace presente el problema de que esta planificación no cumple con la cuota mínima exigida para la leche de tipo C, haciendo falta 20 litros, por lo que en este caso se debería realizar una mezcla de leche en la planta, notando que, se podrían tomar estos 20 litros tanto de lo recolectado por la ruta A o B, o una combinación de ambas. Sin embargo, en este caso, se debería tomar lo faltante de la ruta B, ya que esto implica una menor pérdida de beneficio que tomar la leche de tipo A y degradarla a tipo C. También se puede destacar el hecho de que, en caso de no cumplir la cuota de leche de tipo A, la planificación no podría ser arreglada mezclando leche en la planta procesadora.

Con la mezcla realizada en la planta, el beneficio pasa de 428 UM a 420 UM. Ahora si se considera el coste de transporte de 350 UM, las ganancias totales pasan a ser de 70 UM.

1.1. Modelo matemático

Se presenta un modelo de programación entera mixta (MIP *model*) planteado por Paredes-Belmar [Paredes-Belmar *et al.*, 2016]. Primero se definen los parámetros, luego las variables de decisión, para posteriormente presentar las restricciones y la naturaleza de las variables. Por último, se presenta la función objetivo a maximizar.

1.1.1. Parámetros

En este modelo se considera un grafo completo $G(N, A)$, a continuación se listan sus parámetros:

- A : Conjunto de arcos que representan caminos entre granjas de leche
- A^0 : Conjunto de arcos que representan caminos entre planta y productores de leche
- $N = \{0, \dots, n\}$, Conjunto de nodos que representan los productores de leche
- N_0 : Conjunto de nodos que representan los productores y el nodo $\{0\}$, correspondiente a la planta procesadora
- K : Conjunto de camiones
- T : Conjunto de calidades de leche
- N^t : Conjunto de granjas de leche de calidad $t \in T$
- D^t : Leche de calidad r , resultado de la mezcla de leche de calidad r con leche de calidad t
- IT : Conjunto de pares ordenados (i, t) de granjas i y leche de calidad t , donde cada granja solo produce una calidad de leche
- Q^k : Capacidad de cada camión k
- q_i^t : Cantidad de leche t producida por la granja i
- c_{ij}^k : Costo de viaje de cada camión k sobre el arco $(i, j) \in A \cup A^0$
- α^t : Ingreso por unidad de leche de calidad t
- P^t : Requerimientos de leche de calidad t de la planta
- $K_i = \{k \in K : t \in T, q_i^t \leq Q^k \forall t\}$ Conjunto de camiones que pueden visitar la granja i
- $K_{ij} = \{k \in K : q_i^t + q_j^t \leq Q^k \forall t\}$ Conjunto de camiones que puede viajar de la granja i a la granja j recolectando la leche que ambas granjas producen sin exceder la capacidad del camión, para cada arco $(i, j) \in A \cup A^0$
- 0_k : Nodo en el que el camión k inicia la ruta
- $AK = \{(i, j, k) : (i, j) \in A \cup A^0, k \in K_{ij}\}$ Conjunto de elementos que contiene un arco que va de la granja i a la granja j y un camión k que pertenece a K_{ij}
- PM = Conjunto de pares ordenados (t, r) , donde la leche de calidad t no se puede mezclar con la de calidad r .

1.1.2. Variables

Se definen 5 variables de decisión. Tres de estas son de tipo binaria, mientras que las otras dos tienen la función de controlar la cantidad de leche que entrega el camión a la planta y controlar las mezclas realizadas una vez la leche ha llegado a la planta respectivamente.

- $x_{ij}^k = \begin{cases} 1 & \text{Si el camión } k \text{ viaja directamente del nodo } i \text{ al nodo } j \\ 0 & \text{en caso contrario} \end{cases}$
- $y_i^{kt} = \begin{cases} 1 & \text{Si el camión } k \text{ recoge leche de calidad } t \text{ de la granja } i \\ 0 & \text{en caso contrario} \end{cases}$
- $z^{kt} = \begin{cases} 1 & \text{Si el camión } k \text{ entrega leche de calidad } t \text{ a la planta} \\ 0 & \text{en caso contrario} \end{cases}$
- w^{kt} = Volumen de leche de calidad t que el camión k entrega a la planta
- v^{tr} = Volumen de leche de calidad t entregada a la planta, mezclada para su uso como leche de calidad r

1.1.3. Restricciones

Se tiene un total de 14 restricciones presentadas a continuación:

- La restricción (1) establece que cada camión no puede recolectar más leche que su capacidad máxima.

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

- La restricción (2) establece que la recolección de la leche de cada granja debe ser realizada por exactamente un camión. Es decir, se debe recolectar la leche de todas las granjas y cada granja no puede ser visitada más de una vez.

$$\sum_{k \in K_i} y^{kt} = 1, \quad \forall i \in N, t \in T : (i,j) \in IT \quad (2)$$

- La restricción (3) establece que cada camión debe tener como máximo una ruta, que comienza desde 0_k , que corresponde a la planta procesadora.

$$\sum_{j: (0_k, j, k) \in AK} x_{0_k j}^k \leq 1 \quad \forall k \in K \quad (3)$$

- La restricción (4) permite controlar de flujo para el orden de las visitas de los nodos por parte de cada camión.

$$\sum_{i:(i,j,k) \in AK} x_{ij}^k = \sum_{h:(j,h,k) \in AK} x_{jh}^k \quad \forall k \in K_j, j \in N_0 \quad (4)$$

- La restricción (5) establece que, para cada camión k , si recolecta la leche de la granja i , debe detenerse en esta.

$$\sum_{p:(p,i,k) \in AK} x_{pi}^k = y_i^{kt} \quad \forall k \in K_i, i \in N, t \in T : (i,t) \in IT \quad (5)$$

- La restricción (6) evita que un camión que entrega leche de tipo t a la planta, recolecte leche de una calidad menor a esta.

$$z^{kt} \leq 1 - \sum_{\substack{r \in D^t : r \neq t \\ (i,r) \in IT}} y_i^{kr} \quad \forall k \in K_i, i \in N, t \in T \quad (6)$$

- La restricción (7) establece que cada camión solo entrega un tipo de leche a la planta, pudiendo esta ser resultado de una mezcla.

$$\sum_{t \in T} z^{kt} \leq 1 \quad \forall k \in K \quad (7)$$

- La restricción (8) establece que la leche de cada tipo entregada por cada camión no debe superar su capacidad.

$$w^{kt} \leq z^{kt} Q^k \quad \forall k \in K, t \in T \quad (8)$$

- La restricción (9) mide la cantidad recolectada de cada tipo de leche por cada camión, considerando las granjas que ha visitado para que no supere su capacidad.

$$w^{kt} \leq \sum_{r:i \in D^r} \sum_{h \in N^r} q_h^r y_h^{kr} \quad \forall k \in K, t \in T \quad (9)$$

- La restricción (10) fuerza a que toda la leche producida por las granjas sea llevada a la planta.

$$\sum_{k \in K} \sum_{t \in T} w^{kt} = \sum_{(i,t) \in IT} q_i^t \quad (10)$$

- La restricción (11) equilibra la cantidad de leche de cada calidad que llega a la planta y la cantidad de leche de cada calidad restante después de la mezcla en la planta.

$$\sum_{r \in D^t} v^{tr} = \sum_{k \in K} w^{kt} \quad \forall t \in T \quad (11)$$

- La restricción (12) obliga a que se satisfagan las cuotas de leche de cada tipo.

$$\sum_{t \in T} v^{tr} \geq P^r \quad \forall r \in D^t \quad (12)$$

- La restricción (13) evita la mezcla de leche prohibidas

$$y_i^{kt} + y_i^{kr} \leq 1 \quad \forall (t, r) \in PM; (i, t), (j, t) \in IT \quad (13)$$

- La restricción (14) evita la aparición de sub-ciclos en las rutas de cada camión.

$$\sum_{i \in S} \sum_{j \in S} x_{ij}^k \leq |S| - 1 \quad \forall S \subseteq N, k \in K \quad (14)$$

1.1.4. Naturaleza de las variables

El dominio de las variables presentadas con anterioridad se puede ver a continuación.

- Variable que controla la ruta del camión es de tipo binaria.

$$x_{ij}^k \in \{0, 1\} \quad \forall (i, j, k) \in AK$$

- Variables asociadas a los tipos de leche recolectados y entregados a la planta son de tipo binaria.

$$y_i^{kt}, z^{kt} \in \{0, 1\} \quad \forall i \in N, k \in K_i, t \in T : (i, t) \in IT$$

- Variables asociadas a los volúmenes de leche entregados a la planta y mezclados en esta corresponden a valores no negativos.

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

1.1.5. Función objetivo

La función objetivo corresponde a la maximización del beneficio. Esto corresponde a las ganancias aportadas por la leche recibida en la planta procesadora, menos los costos de transporte de las rutas construidas. Esta sólo considera lo anterior, ya que otros elementos como lo serían los pagos a cada una de las granjas por la adquisición de la leche corresponden a un costo fijo.

$$Z = \text{Max} \sum_{t \in T} \sum_{r \in T} \alpha^r v^{tr} - \sum_{(i, j, k) \in AK} c_{ij}^k x_{ij}^k$$

1.2. Objetivos

1.2.1. Objetivo General

Construir un algoritmo basado en una técnica de búsqueda local para obtener el conjunto de rutas para el problema de recolección de leche con mezcla que optimicen las ganancias.

1.2.2. Objetivos específicos

- Diseñar un algoritmo basado en una técnica de búsqueda local.
- Analizar la relevancia de los componentes más importantes del algoritmo propuesto.
- Comparar el rendimiento del algoritmo propuesto en instancias de distintos tamaños presentes en el estado del arte.
- Examinar el rendimiento del algoritmo en instancias más grandes, es decir, más cercanas a la realidad.

CAPÍTULO 2

ESTADO DEL ARTE

El problema de recolección de leche con mezclas corresponde a una sub-rama de un problema más general, conocido como *Milk Collection Problem* (MCP), que a su vez es considerado variante de *Vehicle Routing Problem*. Estos corresponden a un tipo de problemas de optimización combinatoria con diversas aplicaciones en la industria, cuyos beneficios más obvios están ligados a reducir los costos de transporte.

Para el problema de recolección de leche se han propuesto diversas variantes, cuyas diferencias radican principalmente en considerar un solo tipo de leche o más de uno. En este último caso se consideran factores como si es permitida la mezcla de la leche o el uso de camiones con más de un compartimento para transportar distintos tipos.

Por otro lado, dado que MCP es un caso especial de VRP, otros problemas derivados de este han sido aplicados en MCP y por esta razón las restricciones involucradas en aquellos problemas son heredadas por MCP.

A continuación se presentará el estado del arte en torno al problema de recolección de leche con mezcla, presentando en primera instancia la literatura relacionada a VRP junto con variantes de este, para luego profundizar en los trabajos que abordan la recolección de leche como tal, iniciando con problemas que consideran un sólo tipo de leche, para luego pasar a los que consideran más de uno, pero sin la posibilidad de mezclar los diferentes tipos para por último revisar los trabajos que si consideran la posibilidad de mezcla.

2.1. VRP y variantes

2.1.1. Vehicle Routing Problem

Vehicle Routing Problem (VRP) fue introducido por primera vez por Dantzig y Ramser [Dantzig y Ramser, 1959] en 1959, aunque en ese momento, el problema era denominado como *Truck Dispatching Problem* y corresponde a un problema NP-Hard (por lo que sus variantes también lo serán) ya que puede ser reducido a un problema de vendedor viajero. En este, sus autores describen el problema a través de una aplicación real relacionada con la entrega de gasolina a estaciones de servicio por parte de una flota de camiones. Además, proponen el primer modelo y algoritmo para afrontar el problema con el objetivo de obtener soluciones que sean realistas y muy cercanas a la solución óptima, utilizando heurísticas y meta-heurísticas, ya que métodos de optimización exactos eran inviables de utilizar debido al tiempo de CPU que consumían al abordar instancias más cercanas a las del mundo real, que naturalmente, eran de mayor tamaño.

Este enfoque fue mejorado por [Clarke y Wright, 1964], en donde se profundizó en abordar casos con flotas con distintas capacidades y una gran cantidad de puntos de entrega. En este caso se desarrolló un proceso iterativo que permite una selección rápida de una ruta óptima o cercana al óptimo.

Este es un problema de gran complejidad y que con el pasar de los años ha adquirido mucha popularidad, la que se puede ver reflejada en la gran cantidad de variantes que posee (con ventanas de tiempo, backhauls, etc.) y el lector puede revisar aplicaciones de uso real en la actualidad en [Toth y Vigo, 2002].

2.1.2. Multi-Product Vehicle Routing Problem

Multi-Product Vehicle Routing Problem (MPVRP) [Liu *et al.*, 2008] es una variante de VRP que consta de transportar diferentes tipos de productos en un mismo vehículo. En este, cada tipo de producto tiene un tamaño diferente y los camiones tienen un tamaño de carga dado, mientras que los clientes se encuentran en ubicaciones definidas y son los que establecen la demanda de cada tipo de producto.

La idea de llevar múltiples productos en un vehículo se introduce con el objetivo de reducir costos de transporte, al igual que en el problema de recolección de leche con mezcla, pero no se contempla que estos se mezclen en el mismo compartimento durante el transporte.

2.1.3. Multi-compartment Vehicle Routing Problem

Multi-compartment Vehicle Routing Problem (MCVRP) corresponde a una variante de VRP propuesta por primera vez por Brown y Graves en 1981 [Brown y Graves, 1981] en la que los camiones poseen compartimentos diseñados para contener los productos a transportar según la densidad de estos.

En [Fallahi *et al.*, 2008] se resuelve MCVRP de tres formas. Primero se utiliza una heurística constructiva sin iteraciones de mejora, luego aplican un algoritmo de *Tabu Search* y por último, un algoritmo memético. Este último corresponde a un algoritmo genético combinado con un procedimiento de búsqueda local que es aplicado después de cada etapa de cruzamiento para intensificar la búsqueda.

2.2. Problemas con un tipo de leche

2.2.1. Primeros acercamientos

El *Milk Collection Problem* (MCP) fue abordado por primera vez como una variante de VRP en 1994, por Sankaran y Ubgade [Sankaran y Ubgade, 1994]. En este se resolvió un caso real de 70 granjas en Etah, India, considerando camiones de diferentes capacidades, un sólo tipo de leche y una restricción relacionada a no exceder el límite de tiempo entre cada recolección, dada por las condiciones climáticas de la zona. Para resolverlo se utilizó un *Decision Support System* (DSS) denominado CARS y fueron capaces de generar rutas de recolección que minimizaron los costes de transporte, obteniendo ahorros anuales de \$15,000 USD.

Luego en 1996, [Igbaria et al., 1996], se hace uso de otro DSS llamado *FleetManager* para resolver un problema de enrutamiento de camiones cisterna para recolectar leche en Nueva Zelanda. En este caso el objetivo era minimizar los costos de entrega de leche a la fábrica por medio de construir rutas más baratas.

2.2.2. Two-Period Travelling Salesman Problem

El problema del vendedor viajero simétrico (STSP, *Symmetric Travelling Salesman Problem*) es un TSP en que la distancia entre dos ciudades (o nodos) es la misma en ambas direcciones, teniendo un grafo no dirigido. En [Butler et al., 1997] se implementa una extensión de este problema para resolver una instancia en una granja lechera en Dublín, Irlanda. En los trabajos anteriores se consideraba sólo una visita a cada granja, pero en esta extensión, el problema plantea una situación en la que existen granjas que requieren una recolección diaria, mientras que otras necesitan ser recolectadas cada cierto número de días, además de considerar camiones con múltiples compartimentos. Para resolver este problema se plantea un modelo de programación entera (IP) de STSP para encontrar la ruta que minimice el costo de transporte de la leche. Finalmente se implementa un algoritmo de branch-and-cut para resolver el modelo IP, cuya diferencia con el algoritmo branch-and-bound corresponde a usar planos de corte para ajustar las relajaciones del modelo.

2.2.3. Heterogeneous Fixed Fleet Vehicle Routing Problem

Heterogeneous Fixed Fleet Vehicle Routing Problem (HFFVRP) es una variante de VRP en la que se considera una flota de vehículos con distintas capacidades. En [Tarantilis y Kiranoudis, 2001] se utiliza por primera vez un algoritmo exacto para poder resolver este problema. La instancia a resolver se encuentra ubicada en Grecia, abordando la distribución de leche fresca para una de las empresas más grandes de este país.

El algoritmo planteado en este trabajo se basa en la aceptación de umbrales (*threshold-accepting based algorithm*) para programar la distribución y rutas de los camiones varias veces a la semana. Este es denominado como BATA (*Backtracking Adaptive Threshold Accepting*) que corresponde a un algoritmo metaheurístico de búsqueda estocástica que pertenece a la clase de algoritmos de aceptación de umbral. Cabe aclarar que este problema no construye las rutas para la recolección de la leche, sino de la leche procesada y otros alimentos perecibles a los supermercados.

2.2.4. Traveling Salesman Problem with Time Windows

Traveling Salesman Problem with Time Windows (TSPTW) es una variante de TSP en el que cada ciudad (o nodo) debe ser visitado en una ventana de tiempo dada. En [Prasertsri y Kilmer, 2004] se aborda este problema para determinar si es económicamente factible implementar un enrutamiento y planificación más eficiente en la recolección de leche de una cooperativa en Florida, Estados Unidos. Para resolverlo se implementó un DSS denominado ALR (*ArcLogistics Route 2.0*) que tenía como objetivo minimizar la distancia recorrida y el tiempo de espera en las ventanas de tiempo.

2.2.5. Truck and Trailer Routing Problem

Truck and Trailer Routing Problem (TTRP) es una variante de VRP planteada junto con su modelo de programación entera (IP), por primera vez, en [Chao, 2002], en la que los camiones poseen remolques. En este se plantea que existen clientes que pueden ser atendidos sólo por un camión, mientras que otros por un camión que puede o no estar tirando un remolque [Lin *et al.*, 2009]. En estos, se puede hacer necesario que el camión deje el remolque en la ruta ya que la ubicación geográfica del cliente posee difícil acceso y posteriormente se devuelva a donde dejó el remolque para continuar su ruta. Hoff y Løkketangen [Hoff y Løkketangen, 2008] utilizan este problema para resolver un caso real en Noruega, utilizando un algoritmo de búsqueda Tabú (TS) con el objetivo de minimizar todas las rutas.

2.2.6. Scheduling and Sequencing Problem

En [Basnet *et al.*, 1999] se propone un algoritmo exacto para resolver un problema de planificación y secuenciación (*Scheduling and Sequencing problem*) junto con un procedimiento heurístico para asignar camiones a rutas predefinidas en Nueva Zelanda. El objetivo de este era minimizar el tiempo en el que el último camión entrega su carga a la planta. El algoritmo propuesto utiliza la estructura del algoritmo branch-and-bound para resolver el modelo de programación entera (IP) propuesto. En este se hace relevante el hecho de que, una vez el camión llega a la planta procesadora, los tanques deben ser vaciados por bombas, cuyo número es limitado, por lo que, dadas ciertas situaciones, se podría llegar a formar una cola,

lo que finalmente retrasaría la visita a otras granjas asignadas en un segundo trayecto de recolección.

2.2.7. Annual Dairy Transportation Problem

Annual Dairy Transportation Problem (ADTP) es un problema de asignación y enrutamiento planteado en [Masson *et al.*, 2016] inspirados en un problema de recolección de leche canadiense. Este se basa en el trabajo de dos de sus autores en años anteriores [Lahrichi *et al.*, 2015], en donde plantean el *Dairy Transportation Problem* (DTP). En este último se consideran variaciones en función de la demanda diaria, produciendo ahorros de hasta un 4 % en términos de distancia y se resuelve por medio de un algoritmo *Generalized Unified Tabu Search* (GUTS).

ADTP, a diferencia de DTP, considera información semanal durante el diseño anual de rutas, por lo que, durante cada semana del año, cada planta debe especificar su demanda de leche. Además, la cantidad total de leche que las plantas reciben al final de la semana debe estar cerca de la cantidad solicitada, considerando una tolerancia para cada una. En su trabajo primero se resuelve DTP con un algoritmo de *Adaptive Large-Neighborhood Search* (ALNS) que provee mejores resultados que los presentados con GUTS, luego se procede a resolver ADTP y se usa un algoritmo de agrupamiento (*k-means*) para agrupar las semanas.

2.3. Problemas con más de un tipo de leche, sin mezcla

2.3.1. Algoritmos genéticos

En [Dooley *et al.*, 2005] se plantea resolver una instancia para las islas del norte y sur de Nueva Zelanda, considerando 30 granjas en cada una de estas regiones. Para resolver este problema se utilizó un software de algoritmo genético para buscar el orden en que los camiones debían recolectar la leche de cada granja. En este se consideraban dos tipos de leche y camiones con 2 compartimentos (el camión y el trailer) de distinta capacidad. La solución inicial a ser entregada al software es obtenida a través de una heurística *greedy*, de tal manera que visitasen las que estaban más cerca a la fábrica, para luego moverse a la siguiente más cercana, hasta que se hayan visitado todas. Se generaron seis órdenes de clasificación iniciales, ordenando las paradas por el tipo de leche dentro del orden de visita de las granjas y ordenando las granjas dentro del orden de visita según el tipo de leche, tanto en sentido horario como antihorario. Luego, se crearon 6 órdenes de clasificación en las que se toma en cuenta que los tanques del camión estén lo más llenos posibles antes de retornar a la fábrica.

Por otro lado, en [Sethanan y Pitakaso, 2016], Sethanan y Pitakaso plantean el uso de algoritmos genéticos considerando tres calidades de leche y camiones con tres compartimentos,

uno para cada tipo. En este se busca minimizar los costos relacionados a la limpieza y desinfección de los vehículos, ya que la leche recolectada de distintos centros de recolección no puede ser mezclada y los tanques deben ser limpiados en la fábrica antes de poder ser usados nuevamente. En este se propone una *Differential Evolution metaheuristic* sobre la que se proponen modificaciones para generar 5 algoritmos distintos que incluyen procesos de reencarnación y de supervivencia para mejorar las soluciones. En este trabajo se sugiere considerar la mezcla de leche de diferentes centros de recolección en un mismo compartimento en futuras investigaciones.

En [Chokanat *et al.*, 2019], se considera el *Raw Milk Transportation Problem*. En este, el costo del transporte depende directamente del uso del combustible, que es consumiendo mientras el camión se mueve de un lugar a otro y durante los tiempos de espera cuando los camiones llegan a la fábrica y no pueden transferir la leche en los tanques de producción. En este se propone un algoritmo de evolución diferencial modificado.

2.3.2. Truck and Trailer Routing Problem

En el año 2010, Caramia y Gerriero [Caramia y Guerriero, 2010] resolvieron el TTRP para encontrar las rutas de recolección de leche de 4 calidades sin que estas sean mezcladas y con el uso de camiones con compartimentos. Aplicaron una heurística de búsqueda local en la que, primero, se asigna a cada granja un compartimento dentro de los camiones, con el objetivo de minimizar el número de estos y posteriormente se busca minimizar la distancia recorrida.

2.3.3. Multi-attribute Vehicle Routing Problem

Multi-attribute Vehicle Routing Problem (MAVRP) corresponde a una variante de VRP, en el que, con el objetivo de proveer modelos más representativos de las situaciones que ocurren en el mundo real, se consideran más atributos que llevan a las características, restricciones y objetivos que definen el problema. En [Dayarian *et al.*, 2015b] se tiene como objetivo generar la planificación de rutas para una flota heterogénea de vehículos, en donde cada uno de estos puede partir de diferentes depósitos, pero luego de recolectar el producto deben depositarlo en una planta procesadora común para finalmente volver al depósito en donde comenzaron su ruta. Además, se considera que los camiones deben visitar a los productores en ventanas de tiempo específicas.

En su trabajo no fue posible resolver el problema como un modelo de programación entera mixta (MIP) ya que era demasiado grande, por lo que lo modelan como un *set covering* utilizando el método de generación de columnas. El algoritmo utilizado corresponde a un *branch-and-price*, de la clase *branch-and-bound* que utiliza generación de columnas, para llevar a cabo el cálculo de los límites inferiores. En cada iteración de la generación de columnas se resuelve un problema maestro lineal restringido, en lugar del problema maestro en

sí.

Se destaca que en este se establece que, aunque la leche deba recolectarse todos los días, las variaciones en la producción diaria debido a factores como el clima o enfermedades del ganado son relativamente insignificantes, por lo que las rutas de recolección no tienen que ser reprogramadas diariamente. También se menciona que el modelo más similar a este en la literatura corresponde a *multi-depot heterogeneous vehicle routing problem with time windows* (MDHVRPTW).

2.3.4. Multi-period Vehicle Routing Problem

Multi-period Vehicle Routing Problem (MPVRP) corresponde a una variante de VRP en la que se considera optimizar la recolección de la producción diaria de varios productores dado un horizonte de planificación. En este contexto, se crea una planificación de rutas para todo el horizonte tomando en cuenta elementos como que los suministros de los productores pueden variar dependiendo de la temporada. En el horizonte de planificación existe una secuencia de temporadas o períodos en los que, internamente se ignoran las variaciones que pueden ocurrir.

En [Dayarian *et al.*, 2015a] y [Dayarian *et al.*, 2016] se resuelve un MPVRP para la recolección y distribución de leche en múltiples períodos, considerando variaciones estacionales significativas de producción en el horizonte de planificación, que utiliza heurísticas basadas en VRP, motivados por un problema real de recolección de leche en Quebec, Canadá. En [Dayarian *et al.*, 2015a] se propone un modelo IP utilizando un método generador de columnas y un algoritmo branch-and-price considerando una flota heterogénea y ventanas de tiempo. Por otro lado, en [Dayarian *et al.*, 2016] resuelven este problema a través de un algoritmo de Adaptive Large-Neighborhood Search (ALNS).

2.4. Problemas con más de un tipo de leche, con mezcla

2.4.1. Milk Collection Problem with Blending

En 2016, Paredes-Belmar plantean Milk Collection Problem with Blending (MCwB) [Paredes-Belmar *et al.*, 2016], nombre en inglés del problema ya definido anteriormente. En este se analizan los beneficios de realizar la mezcla, dando como resultado más beneficios para la planta procesadora. Para resolver este problema plantearon un modelo de programación lineal entera mixto y un algoritmo branch-and-cut para resolver instancias pequeñas de hasta 100 nodos (que pueden ser granjas o puntos de recolección) que implementa 3 algoritmos para realizar los cortes.

Sin embargo, para instancias más grandes, el algoritmo propuesto consumiría demasiado

tiempo, por lo que plantearon un procedimiento heurístico de 3 etapas con el objetivo de dividir estas instancias con una mayor cantidad de nodos en subproblemas más pequeños y resolverlos por separado. Estas consisten en:

- Primera etapa: Las granjas se deben separar en *clusters*, proceso que se hace por medio de una heurística *ad-hoc* (de tipo geográfica).
- Segunda etapa: Se asigna a cada cluster una cuota mínima de leche para garantizar que se cumpla lo solicitado por la planta procesadora.
- Se utiliza el algoritmo branch-and-cut propuesto para resolver cada *cluster*.

En este se realizaron pruebas con 40 instancias conocidas de VRP adaptadas para MCwB que tenían un rango de 23 a 101 nodos (incluyendo la planta procesadora) que formaban un grafo completamente conexo, considerando 3 camiones con distintas capacidades de tal manera que fuesen capaces de transportar toda la leche de las granjas. Además, se considera una instancia real con 500 granjas de leche ubicada en la región de Los Lagos, Chile, la que pudo dar resultados favorables dentro de límites de tiempo de 1.65 horas con un procesador de 4 núcleos, lo que lleva a concluir que se diseñó una heurística con un rendimiento en cuanto a tiempo y resultados aceptable, considerando que las rutas de los camiones no deben cambiar diariamente. Sin embargo, los mejores resultados se obtuvieron con tiempos cercanos a 16 horas y dividiendo esta instancia en 13 zonas geográficas diferentes.

A continuación se muestran los resultados para las instancias de prueba pequeñas, en donde:

- Net corresponde a la instancia a resolver.
- $Q[*; *; *]$ indica la capacidad para cada camión en miles de litros.
- $P[*; *; *]$ son los requerimientos de leche de tipo A, B y C respectivamente en miles de litros.
- $|N_0|$ es el número de nodos en la red, incluyendo la planta procesadora.
- Z es el beneficio óptimo en unidades monetarias.
- VA, VB y VC corresponden a la cantidad de leche de cada calidad en litros entregada y mezclada en la planta.
- T corresponde al tiempo transcurrido en segundos por la CPU.
- TB indica si ocurrieron mezclas en los camiones.
- PB indica si ocurrieron mezclas en la planta procesadora.

Las primeras 9 instancias han sido obtenidas del *TSPLib set* [Reinelt, 1991] y son mostradas en la tabla 2. Las siguientes 3 son tomadas del trabajo de Fisher [Fisher, 1994] y se encuentran en la tabla 3. Luego, 6 de estas pertenecen al trabajo Taillard [Taillard, 1999], encontrándose en la tabla 4 y por último 22 son tomadas del trabajo de Augerat [Augerat *et al.*, 1995] y sus resultados se muestran en la tabla 5. En todas estas la producción de cada granja es conocida.

Tabla 2: Instancias obtenidas del trabajo de Reinelt.

Fuente: Paredes-Belmar *et al.*

<i>Net</i>	<i>Q</i>	<i>P</i>	$ N_0 $	<i>Z</i>	<i>VA</i>	<i>VB</i>	<i>VC</i>	<i>T</i>	<i>TB</i>	<i>PB</i>
eil22.vrp	[10;15;20]	[6;5;4]	22	15,947	9,800	7,200	5,500	12	no	no
eil23.vrp	[6;7;8]	[1; 1.5; 2]	23	7,207	6,100	2,009	2,080	6	no	no
eil30.vrp	[5;5.5;6]	[2.2;2.4;2.6]	30	7,117	2,400	6,000	4,350	99	yes	no
eil31.vrp	[55;50;40]	[10;5;8]	31	60,080	28,300	34,800	27,200	66	no	no
eil33.vrp	[20;20;15]	[8;7;6]	33	20,409	11,200	11,690	6,480	58	no	no
eil51.vrp	[25;30;35]	[22;23;24]	51	50,128	22,400	29,600	25,700	154	no	no
eil76.vrp	[45;48;51]	[45;43;40]	76	91,461	46,800	46,700	42,900	1,700	no	no
eil101.vrp	[50;55;60]	[47;48;49]	101	96,115	48,800	48,000	49,000	66,843	yes	yes
att48.vrp	[50,45,40]	[38,34,30]	48	17,452	40,000	37,500	40,000	284	yes	no

Tabla 3: Instancias obtenidas del trabajo de Fisher.

Fuente: Paredes-Belmar *et al.*

<i>Net</i>	<i>Q</i>	<i>P</i>	$ N_0 $	<i>Z</i>	<i>VA</i>	<i>VB</i>	<i>VC</i>	<i>T</i>	<i>TB</i>	<i>PB</i>
f45.vrp	[20;15;10]	[5;10;5]	45	23,705	9,760	18,020	8,320	80	yes	no
f71.vrp	[50;50;50]	[25;20;10]	72	72,864	26,865	49,998	37,977	3,483	yes	no
f72.vrp	[60;60;60]	[20;30;40]	72	72,072	26,861	47,979	40,000	3,659	yes	yes

Tabla 4: Instancias obtenidas del trabajo de Taillard.

Fuente: Paredes-Belmar *et al.*

<i>Net</i>	<i>Q</i>	<i>P</i>	$ N_0 $	<i>Z</i>	<i>VA</i>	<i>VB</i>	<i>VC</i>	<i>T</i>	<i>TB</i>	<i>PB</i>
tai75A.vrp	[25;30;35]	[10;15;20]	76	65,477	21,980	15,000	31,800	4,806	yes	yes
tai75B.vrp	[30;30;30]	[20;25;25]	76	48,238	24,530	25,000	25,000	15,056	yes	yes
tai75C.vrp	[15;20;25]	[5;10;15]	76	25,906	13,085	10,000	24,515	4,537	yes	yes
tai75D.vrp	[20;35;30]	[15;15;20]	76	65,477	20,935	19,880	30,060	2,645	no	no
c50.vrp	[35;30;30]	[15;30;26]	51	49,803	21,700	30,000	26,000	84	yes	yes
c75.vrp	[40;50;50]	[40;45;50]	76	86,677	40,000	46,400	50,000	13,760	yes	yes

También sus autores discuten sobre el uso de camiones con compartimentos en vez de realizar mezcla de leche y sobre si mezclarla o no. Respecto al uso de camiones con compartimentos no se puede concluir si es mejor o peor ya que la dominancia de una alternativa u otra depende del tamaño de los compartimentos y la producción de leche de las granjas, pero en el caso de utilizar camiones con compartimentos se requería un número mayor de estos, además de que los camiones de este tipo tienen un mayor costo. Mientras que respecto a realizar mezcla o no, las soluciones que consideran este procedimiento presentan mejores resultados que las que no. Finalmente. se concluye que la mezcla de leche es un

Tabla 5: Instancias obtenidas del trabajo de Augerat.

Fuente: Paredes-Belmar *et al.*

Net	Q	P	$ N_0 $	Z	VA	VB	VC	T	TB	PB
a32.vrp	[10;15;20]	[12;10;8]	32	26,660	16,200	10,000	14,800	23	no	yes
a33.vrp	[15;20;25]	[15;8;6]	33	29,417	17,600	11,400	15,600	62	no	no
a34.vrp	[20;20;25]	[10;12;14]	34	30,496	15,900	16,000	14,000	40	no	yes
a36.vrp	[20;15;15]	[10;12;14]	36	29,233	16,000	14,200	14,000	110	no	yes
a37.vrp	[20;15;10]	[10;8;6]	37	24,837	10,000	16,200	14,500	45	yes	no
a38.vrp	[20;20;10]	[10;15;15]	38	28,596	10,000	20,000	18,100	570	yes	no
a39.vrp	[20;20;20]	[10;12;14]	39	30,808	14,600	17,300	16,600	110	no	no
a44.vrp	[25;20;15]	[20;16;12]	44	38,771	23,400	16,000	17,600	101	yes	yes
a45.vrp	[25;20;20]	[20;18;18]	45	40,282	23,300	18,000	18,000	136	yes	no
a46.vrp	[30;25;20]	[16;17;18]	46	40,696	22,400	19,900	18,000	66	no	no
a48.vrp	[30;25;20]	[20;20;20]	48	39,800	20,100	20,000	22,500	230	yes	yes
a53.vrp	[30;30;30]	[20;20;20]	53	46,662	23,490	24,870	23,250	183	no	no
a54.vrp	[15;15;15]	[5;5;5]	54	22,414	12,600	11,650	9,200	304	no	no
a55.vrp	[15;15;20]	[5;10;15]	55	24,694	11,900	12,250	17,800	270	no	no
a60.vrp	[20;10;20]	[8;12;16]	60	25,041	11,800	13,650	16,000	3,565	yes	yes
a61.vrp	[35;35;35]	[30;20;10]	61	60,644	30,400	34,600	23,500	561	no	no
a62.vrp	[15;15;15]	[10;11;12]	62	22,917	12,500	11,000	13,150	1,022	yes	no
a63.vrp	[20;20;20]	[5;10;20]	63	24,447	10,050	13,600	20,000	2,930	yes	yes
a64.vrp	[20;20;20]	[5;10;20]	64	24,100	11,750	10,650	20,000	5,395	yes	yes
a65.vrp	[15;15;15]	[10;12;14]	65	28,046	14,350	15,000	14,500	478	yes	no
a69.vrp	[20;20;20]	[10;15;15]	69	25,822	11,750	15,500	15,000	1,552	yes	no
a80.vrp	[20;20;20]	[16;10;16]	80	29,977	16,250	14,650	16,200	5,626	no	no

procedimiento que predomina sobre otros llevando a mejores resultados que reportan mayores beneficios, considerando también el hecho de realizar mezcla cuando se encuentra con camiones de múltiples compartimentos y se propone a futuro incluir puntos de recolección.

Otro trabajo a considerar es el propuesto por Jorge Villagrán en 2019 [Villagrán, 2019], donde propone una heurística basada en *Hill Climbing* para MCwB. En este, las soluciones iniciales se construyen priorizando cumplir con las cuotas que pide la planta procesadora, permitiendo que las visitas asignadas a los camiones puedan exceder su capacidad máxima al realizar la recolección. El algoritmo de *Hill Climbing* propone dos procedimientos para realizar la búsqueda local y corresponden a los siguientes:

- *intra-rutas*: tiene el objetivo de reparar las rutas de los camiones de manera individual, reasignando el orden de visita de las granjas para minimizar los costos de transporte. Este se realiza utilizando el movimiento 2-OPT, que consiste en seleccionar 2 nodos de la ruta y el orden de la sub-ruta presentada entre estos es invertida, lo que permite realizar cambios significativos cuando los nodos seleccionados se encuentran a una distancia significativa.
- *extra-rutas*: tiene el objetivo de reparar las rutas de los camiones en general, reasig-

nando la visita de una granja a cualquier otro camión que provoque que la solución mejore. Este movimiento hace posible que el largo de las rutas varíe.

Estos movimientos hacen necesario que la solución inicial considere la asignación de todas las granjas a la solución. Luego se propone una modificación al procedimiento propuesto, correspondiente a un *Iterated Local Search* (ILS). En un ILS, a diferencia de *Hill Climbing*, no se parte de una nueva solución cuando se llega a un óptimo local, sino que este óptimo se perturba de alguna forma y se parte de este como nueva solución inicial. En este caso la perturbación consiste en cambiar un porcentaje de los nodos de forma aleatoria, sin tener en consideración el cumplimiento de las restricciones o el empeoramiento de la solución. En su trabajo se prueba el rendimiento de sus algoritmos con las mismas instancias propuestas por Paredes-Belmar en [Paredes-Belmar *et al.*, 2016], obteniendo buenos resultados tanto con *Hill Climbing* e ILS, pero siendo mejores con ILS. También se destaca el hecho de obtener mejores soluciones para la instancia de 500 granjas.

2.4.2. Milk Collection Problem with Blending and Collection Points

Milk Collection Problem with Blending and Collection Points (MBCP) [Paredes-Belmar *et al.*, 2017] es una variante de MCwB planteada por los autores de este último [Paredes-Belmar *et al.*, 2016]. En este, a diferencia del original, se considera la inclusión de puntos de recolección y las decisiones de donde ubicar estos, resultando en que la formulación combine un problema de enrutamiento con un problema de asignación y un problema de ubicación. Los puntos de recolección funcionan como puntos en donde los productores de granjas que se encuentran alejadas podrían almacenar la leche que producen, reduciendo así los tramos y nodos que los camiones deben visitar, pasando de, por ejemplo, tener que visitar tres granjas separadas a visitar sólo un punto de recolección en algún punto cercano a estas. Cabe destacar que el introducir los puntos de recolección tiene el propósito de reducir los costos posicionando puntos de recolección equipados con tanques donde los productores deben trasladar la leche producida para la posterior recolección de los camiones siempre que el costo de que el productor realice este traslado sea menor al costo de hacer una recolección directa a la granja.

En este, se plantean algoritmos similares a los de su primer trabajo, pero en el procedimiento heurístico se incluye un algoritmo de optimización de colonia de hormigas, además de considerar la posibilidad de mezclar o no mezclar leche y el utilizar o no puntos de recogida, teniendo 4 diferentes esquemas. Para probar el rendimiento se utilizaron 40 instancias pequeñas y se dividió la instancia de 500 granjas en zonas pequeñas, con 80 camiones y 112 puntos de recolección, sin embargo, los resultados fueron desfavorables obteniendo soluciones de peor calidad y utilizando más de 34 horas en su ejecución, mientras que si no se dividían la heurística obtuvo buenas soluciones en tiempos acotados.

CAPÍTULO 3

PROPUESTA DE SOLUCIÓN

Para resolver el problema de recolección de la leche con mezcla (MCwB), se propone un algoritmo de búsqueda local basado en *hill-climbing* con dos movimientos. Para comenzar, se genera una solución que intentará, dentro de lo posible, generar las rutas para los camiones, de tal manera que se cumpla con las restricciones relacionadas a las cuotas mínimas de cada tipo de leche que pide la planta y no exceder la capacidad de los camiones.

Una vez hecho esto, se procede a buscar una solución mejor a la actual de forma iterativa, por medio de movimientos que buscan reubicar las granjas en las rutas de los camiones y reasignar los camiones a las rutas, hasta obtener el óptimo local de la zona en la que se está buscando.

Por medio de esta propuesta se pretende abordar nuevos elementos del problema en el algoritmo creado, tal como la restricción de movimientos según las calidades de granjas y rutas, o bien, la facultad de intercambiar rutas de recolección entre camiones. También se busca evaluar cómo esto influye en el desempeño del algoritmo (en términos de tiempo de ejecución y calidad de la solución obtenida) frente a situaciones cercanas a la realidad. Se espera que esto sea un aporte para el desarrollo de la investigación en torno a este problema (y problemas similares), de tal manera que sean analizados al momento de construir nuevos algoritmos.

A continuación, se describe en detalle la implementación del algoritmo para resolver el MCwB, abordando la representación utilizada, la función de evaluación y el algoritmo a implementar, comprendiendo la generación de la solución inicial, los movimientos a utilizar y la forma en que se obtienen los vecinos de la solución actual.

3.1. Representación

Dado que en el MCwB se busca encontrar el conjunto de rutas óptimas para un conjunto de camiones que deben visitar un determinado número de granjas, cumpliendo las cuotas mínimas que exige la planta procesadora, se decide representar a la solución como un listado de rutas (igual al número de camiones) que indique el orden en el que se recolectará el producto de las granjas.

Es por esto que en el algoritmo se utiliza un vector de vectores para representar la solución. Se tiene un vector de rutas, en el que cada una de estas corresponde al recorrido de granjas visitadas por un camión. Dentro de cada ruta las granjas se representan mediante un identificador único, dado por el archivo de la instancia que se quiere resolver.

Con esto, si se quiere resolver una instancia con n camiones y m granjas, se tendrán n rutas que en conjunto tendrán m granjas. Una representación de una solución para una instancia pequeña de 3 camiones y 22 nodos (21 granjas y 1 planta procesadora) se puede ejemplificar en la figura 2.

[[20, 5, 2, 11, 8, 14, 17]
[6, 21, 12, 9, 18, 15, 3]
[19, 10, 22, 7, 16, 13, 4]]

Figura 2: Ejemplo de instancia pequeña con 3 camiones y 21 granjas
Fuente: Elaboración propia

Se debe dar cuenta que en las instancias el nodo 0 corresponde a la planta procesadora, de la cual los camiones inician y finalizan su ruta, por lo que en el ejemplo anterior, la primera ruta iniciaría en el nodo 0, iría luego a la granja 20, continuaría su recorrido hasta llegar a la número 17 y de esta volvería al nodo 0. Este nodo se omitirá en la representación, ya que siempre se debería mantener al inicio y final de cada ruta, pudiendo generar conflictos al momento de querer realizar un movimiento y reubicarlo en alguna parte intermedia de la ruta. Al excluirlo, se ahorra espacio de almacenamiento y sólo es considerado el costo en el que incurre salir de la planta hasta la primera granja y volver a ésta desde la última, al momento de calcular la calidad con la función de evaluación.

Si bien, en el ejemplo de la figura 2, la ruta de cada camión contiene exactamente 7 granjas, el número de granjas que puede visitar un camión dependerá netamente de la capacidad de este y la producción que cada granja en su ruta tiene. Entonces, un camión con una gran capacidad podrá tener más granjas en su ruta, en la medida que las granjas productoras que se le asignen generen cantidades de leche menores, ya que, lógicamente, al asignarle productores con una gran capacidad de producción se limitará la cantidad de granjas que podrá visitar, debido a que, un gran porcentaje de la capacidad del camión será consumido por dichas granjas. Por lo anterior, se decide usar un vector para representar cada ruta individual, ya que este permite variar la longitud de las rutas según se necesite, en contraste a una representación matricial, a la cual se le tendría que definir un largo máximo y seguramente, muchas celdas terminarían por no ser ocupadas.

3.2. Función de evaluación

La calidad de cada solución será cuantificada por la función objetivo planteada en el modelo matemático, es decir, las ganancias aportadas por cada tipo de leche entregado a la planta procesadora, menos los costos incurridos por los camiones de efectuar las rutas para recolectar y llevar la leche a la planta, sumado a dos penalizaciones por el incumplimiento de restricciones.

Primero, las ganancias son calculadas ruta por ruta de manera individual, sumando la cantidad de leche que transporta esa ruta y multiplicándola por el ingreso que produce la leche de menor calidad transportada en dicho camión. Luego, los costos corresponden a la distancia euclideana comprendida entre las granjas que se encuentran contiguas en cada ruta, sumado a la distancia de la planta a la primera granja y la distancia de la última a la planta de regreso. Se debe dar cuenta que las distancias tratadas en el problema son simétricas, es decir, es la misma tanto en una dirección como en la opuesta. Para este último punto, dado que constantemente se consultarán distancias entre nodos presentes en la ruta, se prepara una matriz de distancias *distanceMatrix* al momento de leer la instancia a trabajar, que contendrá las distancias de cada granja a otra (o a la planta procesadora) y viceversa, teniendo que calcular dichos valores sólo una vez, limitándose sólo a consultarlos cuando sean necesarios para calcular la distancia recorrida en cada ruta.

En cuanto a las penalizaciones, la primera corresponde al incumplimiento de las cuotas exigidas por la planta procesadora, calculándose como:

$$penalizacionCuota = factorCuota \cdot litrosCuota \cdot ingresosCuota$$

Con esto, se tiene una penalización para el incumplimiento de cualquier tipo de leche, compuesta por un factor (igual para todas las leches) multiplicado por los litros que faltan y por el ingreso que aporta cada litro de la cuota dada. Lo anterior quiere decir que, se incurrirá en una mayor penalización por litro si el beneficio que aporta esta es mayor, siendo la máxima penalización para la leche de tipo A y la menor para la de tipo C.

La segunda penalización corresponde a que, para una determinada ruta, el camión asignado no tenga la capacidad suficiente para poder llevar toda la producción de las granjas, calculándose como:

$$penalizacionCapacidad = factorCamion \cdot capacidadExcedida$$

Esta penalización, a diferencia de la primera, no hace una distinción si es que la capacidad necesaria que falta para cumplir, cubriría una producción de leche de una calidad mayor o menor, sino que se calcula por un factor de penalización (distinto al de la primera penalización) común para todas las rutas multiplicado por la capacidad excedida del camión asignado a esta.

Las penalizaciones anteriores se han incluido con la intención de que, durante la ejecución del algoritmo, se pueda trabajar con soluciones infactibles.

El cálculo de la calidad de la solución queda expresado por:

$$Calidad = ganancias - costoRutas - penalizacionCuota - penalizacionCapacidad$$

Adicional a esto, se debe recordar que existe la posibilidad de hacer el proceso de mezcla de tipos de leche en la planta procesadora, en caso de que llegue a faltar alguna cantidad para un determinado tipo. Lo anterior quiere decir que, primero se debe verificar si cada cuota fue satisfecha y en caso de obtener una respuesta negativa para algún tipo de leche, revisar si un tipo de leche de calidad superior cuenta con excedentes que puedan ser mezclados con esta, revisando desde el tipo de leche de menor calidad (que sea mayor a la que se quiere abastecer) hasta la de mayor calidad, con tal de reducir lo menos posible las ganancias aportadas por cada tipo de leche. Cabe notar que, debido a esta posibilidad, los camiones pueden hacer rutas que no cumplan con las cuotas pedidas por la planta, siempre y cuando no sean del tipo A, es decir de la mayor calidad, ya que se podrá cubrir la demanda faltante con leches de calidad superior, por lo que sólo la leche de mayor calidad debe, obligatoriamente, ser cumplida por las rutas que realicen el o los camiones.

Por lo anterior, antes de calcular la calidad de la solución, se verifica el cumplimiento de cuotas y en caso de que esta no cumpla la cuota de algún tipo de leche, se realiza el proceso de mezcla en la planta procesadora.

3.3. Generación de soluciones iniciales

Para generar la solución inicial, se busca realizar una asignación de las granjas productoras a cada una de las rutas a cargo de un camión.

Al momento de realizar esta asignación, se trata de cumplir con las cuotas solicitadas por la planta procesadora, asignando un camión de manera aleatoria a cada una de las rutas, para posteriormente seleccionar granjas al azar para ser ubicadas en estas. Esta asignación de granjas, se hará de tal manera que, en un mismo camión haya sólo leche de un mismo tipo, además de intentar no exceder la capacidad del camión perteneciente a la ruta. En caso de que ninguna ruta tenga la capacidad suficiente para recibir a la granja, se buscará la ruta con la mayor capacidad disponible que transporte el mismo tipo de leche, incurriendo así, en la menor sobrecarga posible en los camiones.

Por último, para las instancias del estado del arte se generarán 6 configuraciones distintas, a partir de la asignación de granjas a las rutas de los camiones. En todas estas se tienen 3 camiones, por lo que, obligatoriamente, se tendrá que asignar uno de estos a cada tipo de leche.

El pseudocódigo propuesto en el algoritmo 1 muestra el procedimiento para la generación de las soluciones iniciales.

Algorithm 1 Generación de soluciones iniciales

```
1: procedure initialSolutions(instance)
2:   trucks  $\leftarrow$  trucks with capacity, shuffled
3:   farms  $\leftarrow$  farms with production and quality
4:   solution  $\leftarrow$  empty vector of routes
5:   neighborhood  $\leftarrow$  genNeighborhood(totalTrucks, totalFams)
6:   assignedFarm  $\leftarrow$  false
7:   altRoute  $\leftarrow$  null
8:   while i < solution.size() do
9:     assing truck[i] to route[i]
10:  end while
11:  while farms  $\neq$  empty do
12:    assignedFarm  $\leftarrow$  false
13:    farm  $\leftarrow$  select random farm
14:    for route in solution do
15:      if routeMilk  $\neq$  farmMilk and route  $\neq$  empty then
16:        continue
17:      end if
18:      if routeCapLeft  $\geq$  farmProduction then
19:        add farm to route from solution
20:        remove farm from farms
21:        update routeCapLeft
22:        assignedFarm  $\leftarrow$  true
23:        break
24:      end if
25:    end for
26:    if !assignedFarm then
27:      for route in solution do
28:        if routeCapLeft  $\geq$  altRouteCapLeft and
29:          routeMilk == farmMilk then
30:          altRoute  $\leftarrow$  route
31:        end if
32:      end for
33:      add farm to altRoute from solution
34:      remove farm from farms
35:      update routeCapLeft
36:    end if
37:  end while
38:  solutions  $\leftarrow$  generateAlternativeConfigurations()
39:  return solutions
40: end procedure
```

Primero, se recopilan los datos necesarios para generar la solución inicial, a partir de la instancia, es decir, las cuotas de cada tipo de leche, los camiones junto con sus capacidades (desordenados), las granjas junto con su producción y tipo de leche y por último, se crea una variable denominada *solution*, correspondiente al vector de rutas, con una cantidad de rutas igual al número de camiones. En la línea 5, se generarán dos vectores, de tamaño igual al número de posibles vecinos que tendrá cualquier solución, según los movimientos utilizados, en base al número de camiones y el número de granjas, asignándolos a la variable *neighborhood*, procedimiento que se abordará en la sección 3.4.3. Además de lo anterior, se establecen dos variables, *assignedFarm*, para identificar si la granja se pudo asignar a alguna ruta y *altRoute*, para seleccionar el mejor candidato para ubicar una granja, en el caso de que no exista la capacidad disponible para agregar esta en alguna ruta.

En el ciclo *while* comprendido entre las líneas 8 y 10, se asigna a cada ruta un camión del vector de camiones, previamente mezclado, para evitar una asignación según el orden establecido en la instancia.

Posterior a esto, se realiza un segundo ciclo *while* desde la línea 11 a la 37, que se encarga de asignar las granjas a las rutas, de manera que la granja produzca el mismo tipo de leche que la ruta a la que se quiere asignar, siempre y cuando la ruta aún tenga capacidad para llevar la producción de dicha granja. Para llevar esto a cabo, se asigna el valor *false* a la variable *assignedFarm*, se selecciona una granja al azar y se itera sobre las rutas presentes en el vector de soluciones, verificando si la leche que lleva la ruta es distinta a la de la granja y que la ruta no esté vacía, continuando a otro camión en caso de que ambas condiciones sean verdaderas. Si no se continúa al siguiente camión, significa que la leche que transporta la ruta es la misma que la que produce la granja, o bien, la ruta está vacía, por lo que puede agregar la granja a la ruta. Sin embargo, primero se presenta un condicional *if*, en la línea 18, encargado de verificar si la capacidad disponible de la ruta es mayor o igual que la producción de la granja y en caso de serlo, se añade la granja a la ruta, se remueve del arreglo *farms*, se actualiza la capacidad restante de la ruta y se asigna a la variable *assignedFarm* el valor *true*, además de romper el ciclo que itera sobre las rutas, continuando el ciclo *while* para seleccionar otra granja.

En el caso de que, al iterar sobre las rutas, no se pueda agregar la granja a ninguna ruta, debido a que se incumpliría el límite de capacidad del camión asociado a las rutas, cuyo tipo de leche es el mismo que el de las granjas, se accede al condicional *if*, comprendido entre las líneas 26 y 36, ya que el valor de *assignedFarm* continuaría siendo *false*. Cuando esto ocurra, se iterará sobre las rutas de la solución, verificando que la capacidad disponible de la ruta sea mayor a la de *altRoute* y que además transporte el mismo tipo de leche que la granja, para así, en caso de que se cumplan dichos condicionales, poder actualizar el valor de *altRoute* con la ruta actual. Lo anterior se hace con el propósito de identificar la ruta compatible con la mayor capacidad disponible, para obtener una menor penalización debido a la capacidad excedida por el camión de la ruta. Una vez identificado el mejor candidato, se añade la granja a este, se remueve del arreglo *farms* y se actualiza la capacidad disponible de dicha ruta, quedando esta vez, con un valor negativo.

Una vez que se hayan asignado todas las granjas a alguna ruta, en la línea 38, se procederá a generar un arreglo de 6 soluciones idénticas entre sí, en las que variará un parámetro de cada ruta. Este parámetro corresponde a la calidad mínima de leche que puede recibir la ruta en cuestión y puede tomar tres valores, dados por las 3 calidades de leche, 0, 1 o 2 (leche de tipo A, B y C respectivamente). A partir de esto, a cada una de las tres rutas se le puede asignar un tipo de leche mínima, limitando así las granjas que pueden recibir a su ruta, no pudiendo agregar una granja que produce leche de tipo C, si es que la calidad mínima de esta es 0 o 1. En caso de caso de que a todas las rutas se les asigne como tipo mínimo 2, sería igual a aplicar el algoritmo sin esta característica, ya que siempre se admitiría cualquier granja en las rutas.

Lo anterior toma importancia al momento de realizar uno de los dos movimientos que se implementan en el algoritmo propuesto, ya que las granjas a mover no podrán reubicarse a rutas cuya calidad mínima sea mayor que la calidad de las granjas. Entonces, si todos tienen como tipo mínimo el 2, se podrían reubicar a cualquier ruta, evaluar la calidad del vecino generado y ver si este se acepta o no. Justamente lo que se quiere evitar es lo anterior, ya que, si una ruta tiene la “etiqueta” de recibir como mínimo leche de tipo 1 y la granja que se quiere mover a esta ruta produce leche de tipo 2, empeoraría la calidad de la leche transportada, pudiendo, seguramente, provocar que se incumpla con una cuota exigida por la planta y obtener una solución con una peor calidad que la actual. Al consultar este parámetro antes de realizar la reubicación de la granja, se tiene como objetivo evitar las operaciones de mover la granja, evaluar la solución y devolver la granja a su posición, debido a que llevaría a un empeoramiento de esta como se explicó anteriormente.

Teniendo en cuenta lo anterior, para las instancias del estado del arte se tienen tres rutas, cada una encargada de transportar leche de una determinada calidad, es decir, visitar granjas de dicha calidad o superior. A la ruta encargada de la leche de tipo A, se le podrá asignar como tipo mínimo 0, 1 o 2, a la de ruta de tipo B se le podrá asignar como tipo mínimo 1 o 2, ya que sería inconsistente dar la posibilidad de que el tipo mínimo sea 0 siendo que la mayoría de granjas en su ruta serían de tipo 1, dando como resultado que sólo se puedan ubicar en la ruta leches de dicho tipo, pudiendo generar a la larga una ruta de calidad 1 con muchas granjas de tipo 0, debido a que solo podrían salir granjas (de tipo 1 o 0) de la ruta, e ingresar a esta solo granjas de tipo 0, lo que sería, posiblemente, un desperdicio al tener que degradar tanta leche de calidad superior. Por último, la ruta encargada del tipo 2, debido a lo explicado anteriormente, sólo puede tener como tipo mínimo 2.

Con esto se generan 6 posibles configuraciones a probar, presentadas en la tabla 6.

Posteriormente, en base a estas pruebas se definirá la configuración de parámetros para la instancia con 500 granjas y 100 camiones.

Tabla 6: Configuración de soluciones iniciales para instancias del estado del arte

Fuente: elaboración propia

	Ruta encargada de		
	tipo 0	tipo 1	tipo 2
Alternativa 1	0	1	2
Alternativa 2	0	2	2
Alternativa 3	1	1	2
Alternativa 4	1	2	2
Alternativa 5	2	1	2
Alternativa 6	2	2	2

3.4. Algoritmo a implementar

Se implementa un algoritmo basado en *Hill-climbing* con Alguna Mejora para la resolución del MCwB. *Hill-climbing* es una técnica de resolución incompleta en la que, a partir de una solución inicial, se busca mejorar la calidad de esta realizando reparaciones según un movimiento (o movimientos) dado.

La representación usada en este algoritmo corresponde a la presentada en la sección 3.1, en la que se trabaja con un vector de vectores para representar el conjunto de rutas de la solución. Cada vector contiene el identificador de una granja y se omite la planta procesadora, con identificador 0. En la figura 3 se presenta un ejemplo de solución para una instancia de 3 camiones y 22 nodos (uno de estos correspondiente a la planta procesadora).

$$\begin{aligned} &[[15, 18, 21, 6, 3, 9, 12] \\ &[17, 20, 14, 5, 2, 8, 11] \\ &[4, 7, 10, 13, 16, 19, 22]] \end{aligned}$$

Figura 3: Ejemplo de instancia pequeña con 3 camiones y 21 granjas

Fuente: Elaboración propia

La función de evaluación a utilizar es la presentada en la sección 3.2, en la que se considera la adición de dos penalizaciones a la función objetivo planteada en el modelo matemático. Esta se presenta a continuación:

$$Calidad = ganancias - costoRutas - penalizacionCuota - penalizacionCapacidad$$

En esta, *ganancias* corresponden al beneficio aportado por cada litro de leche según su tipo (considerando ambos tipos de mezcla) y *costoRutas* es el costo de transporte de las rutas

construidas. En cuanto a las penalizaciones, *penalizacionCuota* es la relacionada al incumplimiento de las cuotas exigidas por la planta procesadora, mientras que *penalizacionCapacidad* castiga la sobrecarga de algún camión, en el caso de que no tenga la capacidad suficiente para poder llevar toda la producción de las granjas de la ruta.

A continuación, se presentan los movimientos que utiliza el algoritmo y la generación del vecindario de los mismos. Por último, se presenta el funcionamiento del algoritmo basado en *Hill-Climbing* con Alguna Mejora.

3.4.1. Movimiento de reubicación de granja

El primer movimiento que realizará el algoritmo corresponde a reasignar una granja a una nueva ubicación, pudiendo llevarse esta a otra ruta, o bien, moverla a otra posición dentro de la misma, con el objetivo de que la solución mejore su calidad. Este corresponde al movimiento principal del algoritmo, ya que su objetivo es reparar las rutas de la solución actual, logrando esto por medio de reducir los costos de transporte al reubicar una granja en otro punto de la misma ruta, o bien, moviendo la granja a otra ruta de tal manera que se mejore la calidad de la solución. Además, al permitir el movimiento de granjas de una ruta a otra, se permite que se produzca la mezcla de leche en estas, ya que las soluciones iniciales sólo construyen rutas cuyas granjas producen el mismo tipo de leche.

En el algoritmo 2 se presenta el funcionamiento del movimiento de reubicación de granja.

Este algoritmo recibe la solución actual, junto con su calidad y el par de posiciones de origen y destino, aún siendo estos códigos correspondientes a números enteros que necesitan ser convertidos a dos números, correspondientes a una ruta y una ubicación dentro de esta.

Primero, se establecen las variables necesarias para realizar el movimiento, se obtiene la posición de origen y de destino de la granja. Además, se necesita de dos variables de tipo entero, para almacenar el índice de la ruta en la que se encuentra el origen y el destino de la granja, una variable para almacenar la granja y una variable booleana para determinar si existió un movimiento de intercambio de camiones.

Se ocupa la función *searchOriginPosition()* para, a partir del código correspondiente a la *farmOrigin*, obtener la referencia de la granja a mover y actualizar los valores de dicha variable y *routeOrigin*, almacenando así la ubicación original de la granja en *solution*. Luego, con la función *getDestinyPositions()* se obtiene el identificador de la ruta de destino y la posición dentro de esta, en la que finalmente se ubicará la granja a mover, a partir del código correspondiente a *farmDestiny*. Una vez ubicada la granja y su ruta de destino, se debe verificar que el tipo de la granja sea compatible con el tipo mínimo asignado a la ruta en la que se ubicará, lo que se realiza en el condicional *if*, que se encuentra en la línea 10, que en caso de tener una calidad inferior que el mínimo aceptado por la ruta, el algoritmo retornará *false*, ya que hasta el momento no se ha realizado ninguna modificación a la solución. Por otro lado, en el

Algorithm 2 Movimiento de reubicación de granjas

```
1: procedure movFarm(solution, currentQlt, positions)
2:   farmOrigin  $\leftarrow$  positions.first
3:   farmDestiny  $\leftarrow$  positions.second
4:   routeOrigin  $\leftarrow$  0
5:   routeDestiny  $\leftarrow$  0
6:   farmToMove  $\leftarrow$  null
7:   capChanged  $\leftarrow$  False
8:   farmToMove  $\leftarrow$  searchOriginPosition(solution, farmOrigin, routeOrigin)
9:   farmDestiny, routeDestiny  $\leftarrow$  getDestinyPositions(solution, farmDestiny)
10:  if farmToMove.milkType > routeDestiny.milkType then
11:    return False
12:  end if
13:  takeOutFarm(routeOrigin, farmOrigin)
14:  putInFarm(routeDestiny, farmDestiny, farmToMove)
15:  newQlt  $\leftarrow$  getQuality(solution)
16:  if capAvailable of truck at routeDestiny < 0 then
17:    capChanged = movCapacity(solution, newQlt, routeDestiny)
18:    if capChanged then
19:      newQlt = getQuality(solution)
20:    end if
21:  end if
22:  if newQlt > currentQlt then
23:    return true
24:  end if
25:  takeOutFarm(truckDestiny, farmDestiny)
26:  putInFarm(truckOrigin, farmOrigin, farmToMove)
27:  return false
28: end procedure
```

caso de que no exista problema con el tipo de leche, se extrae la granja de la ruta de origen con *takeOutFarm()* y se agrega a la ruta de destino con la función *putInFarm()*, calculando finalmente la calidad de la nueva solución y siendo esta almacenada en la variable *newQlt*.

Se continúa al condicional *if*, que abarca desde la línea 16 a la 21, que se encargará de verificar si es que la adición de la granja a la ruta ha sobrepasado la capacidad del camión. En caso de que la capacidad disponible sea negativa, se realizará el movimiento de intercambio de camiones con la función *movTruck()*, que recibe como parámetros la solución, la nueva calidad de la solución (considerando el movimiento de la granja) y la ruta cuyo camión se deberá intercambiar. Al igual que en este movimiento, *movTruck()* modifica la solución actual y la evalúa, retornando *True* si es que la reubicación obtuvo una calidad mayor, o *False* en el caso contrario.

Luego, se evalúa qué valor obtuvo la variable *capChanged*, según la función *movTruck()*, si el intercambio de camiones fue exitoso, se actualiza el valor de *newQlt*, ya que existió un intercambio de camiones, que provocó un aumento en la calidad de la solución, respecto a la que se tenía antes. Si el valor obtenido es *False*, significa que ningún intercambio de camiones pudo mejorar la solución y la solución mantiene el valor de *newQlt*.

Finalmente, desde la línea 22 en adelante, se evalúa si la calidad obtenida con los movimientos realizados es mejor que la anterior, retornando *True* en caso de que se haya obtenido una mejora, o bien deshaciendo el movimiento en caso de que no y retornando *False*.

Se debe dejar en claro que el algoritmo trata la solución por referencia, por lo que no es necesario retornar la solución en ninguna parte del algoritmo. Sólo se pasa esta como parámetro al movimiento de intercambio de camiones para que, al igual que este, pueda modificarla en caso de ser necesario.

3.4.2. Movimiento de intercambio de camiones

El segundo movimiento, encargado de realizar una reasignación de camiones a las rutas, tiene como objetivo reparar una asignación desfavorable de los camiones encargados de transportar la leche dada por la solución inicial. Al generar la primera solución, puede darse, por ejemplo, que la ruta que llevará la leche de mayor calidad tenga el camión con menor capacidad, pudiendo finalmente llevar menos leche de este tipo, ya que los productores deberán ser asignados a otras rutas en donde su calidad se degradará.

Es por esto que este movimiento plantea un intercambio de los camiones que están actualmente asignados a cada ruta, haciendo uso del segundo vector de códigos de movimientos (explicado en la sección 3.4.3), intentando obtener una solución que no incumpla el límite de leche que cada ruta necesita. Este es ejecutado luego de que se haya realizado un movimiento de reubicación de granja, por lo que intentará mejorar la solución modificada por dicho movimiento y obtener una calidad mejor que esta, sin considerar a la solución sin la

reubicación de granja.

En el algoritmo 3 se presenta el funcionamiento de este movimiento.

Algorithm 3 Movimiento de intercambio de camiones

```
1: procedure movTruck(solution, currentQlt, overloadedRoute)
2:   improvement  $\leftarrow$  false
3:   routeDestiny  $\leftarrow$  0
4:   neighborhood  $\leftarrow$  shuffleNeighborhood
5:   for routeToTrade in neighborhood do
6:     if routeToTrade  $\geq$  overloadedRoute then
7:       routeDestiny = routeToTrade + 1
8:     else
9:       routeDestiny = routeToTrade
10:    end if
11:    improvement = swapTruck(overloadedRoute, routeDestiny, currentQlt)
12:    if improvement then
13:      return true
14:    end if
15:  end for
16:  return false
17: end procedure
```

Los parámetros de este algoritmo corresponden a la solución actual junto con su calidad y el identificador de la ruta, cuyo camión no tiene la capacidad suficiente para transportar la leche de todas las granjas que se encuentran en la ruta.

Primero se establecen tres variables para realizar el movimiento, una variable booleana *improvement*, para verificar si se ha conseguido una mejora con el intercambio de camiones, una variable para mantener el identificador de la ruta con la que se intercambiará el camión y una correspondiente al vector de códigos de posibles movimientos, que se traducirán a rutas con las que se intentará intercambiar.

Se realiza un ciclo *For* desde la línea 5 a la 15, iterando en el vector de códigos de movimientos que generan los vecinos de la solución actual (que es explicado en la sección 3.4.3), siendo tratados como identificadores de rutas con las que intercambiar. Dentro de este ciclo se establece el identificador real de la ruta con la que se debe intercambiar, ya que si, por ejemplo, en una instancia con 3 rutas, la segunda es la que está sobrecargada, los identificadores del vector deberían ser 0 y 2, correspondientes a la primera y tercera ruta, pero, debido a que el vector de posibles movimientos se crea una sola vez y se reutiliza, en este caso, tendría los valores 0 y 1, teniendo que hacer un reajuste en el caso de que el identificador en el vector sea mayor o igual al identificador de la ruta sobrecargada. Este consiste en un corrimiento de una posición a la derecha para todos los identificadores establecidos. Una vez detectado el caso, se asigna el identificador a la variable *routeDestiny*.

Con esta información se tiene lo necesario para realizar el intercambio de camiones entre dos rutas. Esto se hace por medio de la función *swapTruck()*, que recibe como parámetros las dos rutas que se intercambiarán los camiones y la calidad de la solución actual. En esta se intercambia el camión asignado a cada ruta, actualizando la nueva capacidad máxima y disponible según el nuevo camión y evaluando la nueva calidad obtenida, retornando *True* en caso de que, debido al intercambio, se obtenga una mejora de calidad, o bien, restaurando la solución a su estado anterior y retornando *False*.

Luego, se asigna a la variable *improvement* el retorno de la función *swapTruck()* y en el caso de que este tome el valor de *True*, se retornará *True* de inmediato, deteniendo la evaluación de otros vecinos. En el caso contrario, se continúa al siguiente vecino y en caso de revisar todos y no obtener ninguna solución vecina, con mejor calidad que la actual, se termina por retornar *False*.

3.4.3. Generación y codificación de los movimientos

Cada solución contiene una serie de elementos. Entre estos se encuentra la ruta que realiza cada camión, teniendo que guardar referencias a cada granja presente (para después poder evaluar la calidad de las mismas), las referencias a cada camión asignado a cada ruta (para recuperar la capacidad máxima de estas), la cantidad y calidad de leche que estas transportan y además, la calidad mínima que cada ruta puede recibir.

Debido a lo anterior, sumado a que la cantidad de vecinos que se pueden generar en base al movimiento de reubicación de granja (presentado en la sección 3.4.1) puede llegar a ser bastante grande, es poco viable y un malgasto de recursos el tener que, en cada iteración, generar el vecindario completo para poder posteriormente mezclar el orden de evaluación de los vecinos.

Entonces, se busca que la representación de los vecindarios sea compacta, evite la construcción de vecinos que no se evaluarán (debido a que el algoritmo propuesto avanzará a la siguiente iteración con el primer vecino que mejore la calidad de la solución actual) y además, permita seleccionar estos de manera aleatoria.

Para el movimiento de reubicación de granja, la mezcla es necesaria para evitar una reparación de la solución de forma determinista a partir de una solución inicial, ya que de esa manera, siempre se repararía la solución evaluando primero los vecinos generados a partir de mover la primera granja, de la primera ruta, a la posición que se encuentra después de la segunda granja, luego después de la tercera y así sucesivamente, ubicándola en todas las posiciones posibles en cada ruta, para luego continuar con la segunda granja, con la que se seguiría el mismo proceso hasta generar los últimos vecinos, producto de mover la última granja de la última ruta.

Lo anterior produciría un comportamiento del algoritmo en el que constantemente se inten-

taría mejorar la solución, moviendo las primeras granjas a todas las posiciones posibles, lo que no es deseable ya que se quiere que cada vecino, o en este caso, tipos de vecino (determinado según la granja a partir de la que fueron generados) tengan la misma oportunidad de ser evaluados, cosa que no ocurrirá si no se mezcla el vecindario, ya que claramente los vecinos generados a partir de mover las últimas granjas de la última ruta serán probados sólo después de que todas las granjas anteriores estén en la mejor posición en la que pueden estar, sin haber considerado las granjas que se encuentran después de ellas. Este problema se presenta debido a que el algoritmo *Hill-climbing* planteado trabaja bajo el esquema de Alguna Mejora, en el que es importante que el vecindario se evalúe de una manera no secuencial, dado que no se revisará completamente para encontrar al mejor vecino como lo sería si se implementase la técnica incompleta con Mejor Mejora, ocurriendo esto último sólo en el caso de que se obtenga un óptimo local.

Teniendo en cuenta lo anterior, se pueden considerar opciones, como primero seleccionar de manera aleatoria una granja o una posición en una determinada ruta y mover esta a todas las posiciones posibles en todas las rutas de manera aleatoria. Realizando esto de tal forma que la granja seleccionada y las posiciones a las que irá no se repetirán, lo que podría ser conseguido con un vector de identificadores de granjas y otro de identificadores de posiciones, realizando una mezcla de estos cada vez que sea necesario, es decir, al inicio de cada iteración para el vector de granjas y para el vector de posiciones, luego de haber elegido una granja a mover. Sin embargo, a pesar de evitar tener que generar el vecindario como tal con esta manera de realizar el movimiento, aún existe cierta secuencialidad al momento de evaluar a los vecinos, ya que una vez que se toma una granja, se deben evaluar todas las soluciones que involucran moverla, antes de avanzar a las soluciones que se producirían al mover la siguiente en el vector de granjas, por lo que aún no se podrían evaluar los vecinos como si se hubiese generado todo el vecindario y luego haber mezclado el orden de este para evaluarlo.

Entonces, dado que se desea obtener una representación del vecindario compacta y que posibilite una evaluación de los vecinos que no sea secuencial, o no siga algún patrón, se ideó una codificación que sea útil para cualquier solución actual, para que no se generen vecinos que no se evaluarán. Esta codificación permite recuperar la granja a mover y la posición a la que se debe mover, en base a la solución actual y el número de granjas y camiones.

Primero, se debe tener en cuenta que el tamaño del vecindario para una determinada instancia, dado el movimiento de reubicación de granja, será siempre el mismo, sin importar la solución con la que se esté trabajando actualmente. Lo anterior, quiere decir que, para este movimiento, la cantidad de vecinos que se generan de mover una granja a otra posición es el mismo, ya que el número de granjas es una constante, como la cantidad de posiciones a las que puede ir, esto último determinado por el número de granjas y camiones.

A continuación, se muestra un ejemplo a través del que se explica la obtención del tamaño del vecindario. En este se presenta una instancia con 21 granjas, enumeradas del 2 al 22 (debido a que el primer nodo enumerado corresponde a la planta procesadora) con 3 rutas confeccionadas de la siguiente manera:

20	5	2	11	8	14	17
----	---	---	----	---	----	----

6	21	12	9	18	15	3	19
---	----	----	---	----	----	---	----

10	22	7	16	13	4
----	----	---	----	----	---

Figura 4: Ejemplo de instancia pequeña con 3 camiones y 21 granjas
Fuente: Elaboración propia

En esta situación se necesita ser capaz de poder seleccionar cualquier granja para ser movida a otra posición. Para esto, lo realmente importante no es saber qué granja se va a mover, sino la posición de la cual se tomará dicha granja, por lo que, para este caso se tienen 21 posibles posiciones, enumeradas del 0 al 20.

↓	↓	↓	↓	↓	↓	↓	
0	1	2	3	4	5	6	
↓	↓	↓	↓	↓	↓	↓	↓
7	8	9	10	11	12	13	14
↓	↓	↓	↓	↓	↓		
15	16	17	18	19	20		

Figura 5: Posiciones posibles enumeradas en una instancia de 21 granjas
Fuente: Elaboración propia

Notar que el número de posiciones no depende de la cantidad de camiones, sino que sólo del número de granjas existentes. En caso de confeccionar las rutas de otra manera, seguirán existiendo sólo 21 posiciones de las cuales tomar una granja, cambiando sólo la ruta en la que se podría encontrar dicha posición, ya que, por ejemplo, en una solución en la que cada ruta tiene asignada 7 granjas, la posición 14 pasaría a ser la posición en la que se encontraría la primera granja de la tercera ruta.

Una vez conocido el número de posiciones de las que se puede elegir una granja, falta determinar el número de sitios a los que esta puede ir. Estas inserciones se realizan al inicio de una ruta, entre dos granjas, o bien al final de esta. Continuando con el ejemplo, se tiene un número igual a 24 posiciones posibles, sin embargo, se deben descontar 2 de estas, correspondientes a las que están contiguas a la granja que se quiere mover, ya que en caso de mover la granja que se encuentra en la posición 2 luego de la primera posición, o antes de la tercera (destacadas en rojo en la figura 6), lógicamente se obtendría la misma ruta, por lo que se debe evitar contabilizar estos como movimientos válidos.

Luego, se puede notar que en cada ruta se cuenta con espacios para insertar granjas igual al número de granjas en su ruta más uno, debido a que, además de poder poner una granja antes de cada una de las que se encuentra en dicha ruta, existe la posibilidad de ponerla después de la última granja. Así, en el ejemplo dado si se quiere mover cualquier granja de

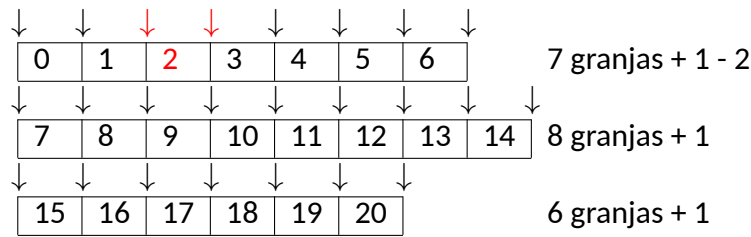


Figura 6: Inserciones posibles en las rutas de una instancia
Fuente: Elaboración propia

la primera ruta a la segunda, se tienen 9 opciones, para moverla a la tercera se tienen 7, mientras que para moverla a otra parte de la misma ruta pasan a ser 6 (8 posibles lugares a los que se les descuentan los adyacentes a la granja). Con esto, la cantidad de posiciones disponibles a las que se puede mover una granja, que se encuentra en alguna posición, va a corresponder a la suma de las granjas de cada ruta más una, pero restando dos debido a que se debe evitar reubicar en las posiciones que obtienen la misma solución, lo que es, finalmente, el número de granjas totales, sumado al número de rutas (o camiones) totales menos 2 unidades.

Así, dado que se sabe la cantidad de posiciones de las que se puede sacar una granja y la cantidad de nuevas posiciones a las que puede ir, el tamaño del vecindario corresponde a:

$$\text{tamañoVecindario} = \text{numGranjas} \cdot (\text{numGranjas} + \text{numRutas} - 2)$$

Siendo igual a 462 en el caso del ejemplo abordado. Sabiendo esto, al momento de leer la instancia a trabajar se generará un vector de enteros, donde estos corresponden a códigos comprendidos entre 0 al tamaño del vecindario menos uno (hasta 461 en el ejemplo). Este vector corresponderá al total de posibles vecinos que se tendrían al momento de realizar el movimiento de reubicación de granja. Este es generado en el algoritmo 1 en la función *genNeighborhood(totalTrucks, totalFarms)* y posteriormente será ocupado en el algoritmo 4, siendo mezclado al inicio de cada iteración, para posteriormente, iterar sobre este para obtener un código.

Luego, en el algoritmo 4, también se debe traducir dicho código a las dos posiciones, la primera siendo de dónde se sacará la granja y la segunda dónde se ubicará. Estos datos son calculados de la siguiente manera:

- $\text{ubicacionGranja} = \text{código} / (\text{numGranjas} + \text{numRutas} - 2)$
- $\text{destinoGranja} = \text{código} \% (\text{numGranjas} + \text{numRutas} - 2)$

Se debe dar cuenta de que estas posiciones aún son globales en la solución, ignorando la estructura de vector de rutas correspondiente a la solución, así que, en el ejemplo dado, si

ubicacionGranja es igual a 10, posteriormente, al momento de realizar el movimiento debe traducirse a *solution[1][3]* y si *destinoGranja* fuese igual a 17, debe traducirse a la posición *solution[2][2]*. Ambas transformaciones pueden hacerse fácilmente conociendo el largo de cada una de las rutas.

De esta manera, se tiene una representación compacta de los movimientos que generan el vecindario, que debe ser construida sólo una vez y podrá ser reutilizada para la misma instancia sin importar el estado actual de la solución, es decir, cuántas granjas se encuentren en cada ruta y en qué posición. Además, al poder mezclar el vector de códigos de movimientos, se evita que la evaluación de vecinos sea secuencial de alguna manera, pudiendo primero realizar un movimiento que mueva la tercera granja de la primera ruta, a alguna posición en la tercera y luego, en caso de que este no mejore la calidad de la solución, realizar otro que mueva una granja ubicada en la segunda o tercera ruta, a cualquiera de las demás, no siguiendo un patrón definido.

Por otro lado, el movimiento de intercambio de camiones (presentado en la sección 3.4.2) es usado en caso de que el movimiento de reubicación de granja resulte en una sobrecarga de la ruta de destino de la granja. Este utiliza un vector construido de manera similar al del movimiento de reubicación de granja. Dicho vector, que contiene los posibles movimientos, será generado al momento de leer la instancia a trabajar y se construirá como un vector con un tamaño igual a la cantidad de rutas menos 1, ya que no se puede intercambiar el camión de la misma ruta consigo mismo.

A continuación, se muestra un ejemplo de la obtención del tamaño del vecindario y el uso del código del vector para este movimiento. En este caso, se considerará una instancia con 5 rutas, cada una de estas con un camión con una capacidad determinada, obteniendo la siguiente situación:

Ruta	1	2	3	4	5
Camión	3	4	2	1	5
Capacidad	3000	3000	3000	4000	3500

Figura 7: Ejemplo de instancia con 5 rutas y camiones
Fuente: Elaboración propia

En este caso, luego de reubicar una granja en otra ruta, se podría llegar a que, por ejemplo, al llevar una granja de la cuarta ruta a la tercera, la cantidad de leche que se tiene en la tercera ruta supere las 3000 unidades que posee el camión 2 asignado a esta, por lo que, antes de evaluar la calidad de dicha solución, se intentará corregir el exceso de capacidad intercambiando el camión asociado a la ruta con problemas, con alguno de otra ruta.

Para esto, se debe dar cuenta que se puede intercambiar con los camiones de las otras cuatro rutas, por lo que, para este vecindario, el vector que guarda los códigos relacionados a la ruta con la que se intercambiará camión tendrá números enteros, comenzando desde el 0 hasta el tamaño del vector, correspondiente a la cantidad de rutas menos uno. En la situación

anterior, el vector tendría cuatro números (del 0 a 3) y se tiene que mezclar cada vez que se quiera intercambiar el camión de una ruta con algún otro, de tal manera que no se prueben intercambios en orden, probando con la primera ruta, luego la segunda y así sucesivamente.

En la figura 8 se puede ver qué código correspondería a cada ruta, destacando en rojo la ruta con el camión cuya capacidad no es capaz de transportar toda la leche producida por las granjas de la ruta. Entonces, al elegir, por ejemplo, el código 3 del vector, se intercambiarían el camión 2 y 5, quedando en las rutas 5 y 3 respectivamente.

Código	0	1		2	3
	↓	↓		↓	↓
Ruta	1	2	3	4	5
Camión	3	4	2	1	5

Figura 8: Rutas con las que intercambiar según el vector de códigos
Fuente: Elaboración propia

De esta manera, se obtiene una representación compacta de todos los posibles movimientos de intercambio de camiones que generan el vecindario, siendo útil para cualquier solución a la que se le quiera aplicar este movimiento. Esta será generada al inicio del algoritmo y se reutilizará cada vez que sea necesario, realizando una mezcla de todos los elementos del vector, evitando así una ejecución secuencial de los movimientos.

3.4.4. Hill-Climbing

La propuesta a implementar corresponde a un algoritmo basado en *hill-climbing* con Alguna Mejora, que hace uso de los movimientos presentados en las secciones 3.4.1 y 3.4.2, además de los vectores para los vecindarios de ambos movimientos cuya generación es explicada en 3.4.3.

Este, a partir de una solución inicial, busca mejorar la calidad de esta al repararla según un movimiento (o movimientos) dado. Para esto, en cada iteración se debe generar un vecindario de soluciones, determinado por la aplicación de un movimiento a la solución actual. Se debe dar cuenta de que, dado que la versión a implementar es con Alguna Mejora, el primer vecino que obtenga una calidad mayor que el actual, será la siguiente solución a la cual se le tendrá que generar el vecindario, avanzando a una siguiente iteración en la que se repetirá el procedimiento hasta que, en algún momento ocurra alguna de las siguientes situaciones:

- Se revisen todos los vecinos de la solución actual: En este caso se habrá llegado a un óptimo local, ya que ninguno habrá superado la calidad de este.
- Se cumpla una ventana de tiempo determinada: En este caso no se habrá llegado al óptimo local, sino a una solución intermedia. Esto puede ocurrir debido a que, a medida

que las instancias crecen en complejidad, se necesita un mayor número de iteraciones y tiempo para poder llegar a un óptimo local, siendo difícil poder obtenerlo en un tiempo acotado.

- Se cumpla un número de iteraciones preestablecido: Al igual que en el caso anterior, no se ha llegado a un óptimo local y se obtiene una solución intermedia.

En caso de querer obtener alguna otra solución, se debe iniciar el algoritmo con otra solución inicial, o bien, si se tuvo que terminar el algoritmo de forma prematura, extender la ventana de tiempo o el número de iteraciones máximo asignado a su ejecución.

Se debe tener presente que el vecindario para cualquier solución de una instancia dada, en este caso, será generado junto con la solución inicial, como se explica en la sección 3.4.3 y en cada iteración sólo será mezclado.

En el algoritmo 4 se muestra el pseudocódigo de la implementación.

Algorithm 4 HC-AM Recolección de leche

```
1: procedure hillClimbing(initRoute, maxTime, maxIter, fCuotas, fCapacidad)
2:   solution  $\leftarrow$  solutions[initRoute]
3:   solImprovement  $\leftarrow$  true
4:   iter  $\leftarrow$  0
5:   while solImprovement and iter < maxIter and time < maxTime do
6:     solImprovement  $\leftarrow$  false
7:     neighborhood  $\leftarrow$  shuffleNeighborhood
8:     currentQlt  $\leftarrow$  getQuality(solution)
9:     for code in neighborhood do
10:      positions  $\leftarrow$  codeToMovement(code)
11:      solImprovement  $\leftarrow$  movFarm(solution, currentQlt, positions)
12:      if solImprovement then
13:        break
14:      end if
15:    end for
16:    iter ++
17:  end while
18:  return solution
19: end procedure
```

Los parámetros que recibe son el número máximo de iteraciones a ejecutar (*maxIter*), el tiempo límite de ejecución (*maxTime*), ambos factores de penalización establecidos en la función de evaluación (*fCuotas* siendo el que penaliza el incumplimiento de alguna cuota y *fCapacidad* para penalizar la sobrecarga que el camión de alguna ruta transporta, debido a que la producción de las granjas excede su capacidad) y además, un índice (*initRoute*) que determinará cuál de las configuraciones de solución inicial se usará durante la ejecución del algoritmo.

En el inicio del algoritmo se establece la solución inicial a probar y se fijan dos variables. La primera, *sollImprovement*, para verificar que luego de realizar un movimiento se haya encontrado una mejora, tomando el valor *True* en caso de que el movimiento haya mejorado la solución con respecto a la actual y tomando el valor *False* en el caso contrario, mientras que la segunda, *iter*, se usa para contabilizar el número de iteraciones realizadas por el algoritmo.

Luego, desde la línea 5 hasta la 17, se realiza el ciclo *while* encargado de realizar la búsqueda local, explorando de forma iterativa el vector de posibles movimientos que genera el vecindario de la solución actual, modificando la solución actual según el movimiento y evaluando el vecino generado, hasta que ninguno de los vecinos sea mejor que esta, se cumpla el número máximo de iteraciones, o bien, se llegue al límite de tiempo de ejecución. En cada iteración se asigna a la variable *sollImprovement* el valor *False*, se mezcla el arreglo *neighborhood* correspondiente a los códigos, que posteriormente se traducirán en movimientos de reubicación de granja y se asigna a la variable *currentQlt* la calidad de la solución actual por medio de la función *getQuality()*.

Luego, en el ciclo *for* comprendido entre las líneas 9 a 15, se llevará a cabo la generación de cada vecino a partir de un *code*. Primero la variable *positions*, correspondiente a un par (<posición de granja>, <destino de granja>) se obtiene a partir de un código por medio de la función *codeToMovement(code)*. Con esto se actualizará el valor de *sollImprovement* según la función *movFarm()*.

La función *movFarm()* recibe como parámetros la solución actual, a la que se le aplicarán los movimientos, la calidad de esta y las posiciones necesarias para mover una granja a una nueva posición. En este proceso se modifica la solución actual junto con su calidad, se evalúa a esta haciendo uso de la misma función *getQuality()* y finalmente retornará *True* o *False* si es que hubo o no hubo mejora respectivamente.

Después de haber evaluado a un vecino, se verifica el nuevo valor de *sollImprovement* y en caso de haber existido mejora, se rompe el ciclo *for*, para finalmente contabilizar la iteración realizada y avanzar a la siguiente. En caso de no mejorar, se debe continuar con el siguiente código de *neighborhood*, hasta que se agoten, momento en el que se ha llegado a un óptimo local, identificando esto porque el valor de *sollImprovement* se ha mantenido como *False*, finalizando con el ciclo *while* y retornando la solución obtenida.

CAPÍTULO 4

VALIDACIÓN DE LA SOLUCIÓN

Para poder evaluar el rendimiento del algoritmo propuesto, se plantean una serie de experimentos. Se definen los objetivos del proceso de experimentación, se analizan las instancias con las que se trabaja y se describe el equipo en donde se realizan las pruebas. Luego se exponen los resultados obtenidos y se realiza una comparación con los presentados en el estado del arte.

4.1. Objetivos

A través del proceso de experimentación se desarrollarán los siguientes objetivos:

- Analizar la relevancia de los componentes más importantes del algoritmos propuesto. Para llevar a cabo esto se experimentará con valores para las penalizaciones propuestas en la función de evaluación, correspondientes al incumplimiento de cuotas o sobrecarga de camiones y se evaluará el comportamiento de los dos movimientos propuestos. Para el movimiento de reubicación de granja, se comparará con las diferentes configuraciones iniciales del tipo de leche mínima que pueden recibir las rutas, comparando cuántos movimientos o vecinos se evalúan y los resultados que se obtienen, mientras que para el movimiento de intercambio de camiones se analizará la importancia de poder reasignar los camiones a medida que se ejecuta el algoritmo, contabilizando las veces que efectivamente esto se hace y significa una mejora en la solución, además de realizar pruebas sin este, evaluando la calidad de las soluciones obtenidas.
- Comparar el rendimiento del algoritmo propuesto, en instancias de distintos tamaños presentes en el estado del arte.
- Examinar el rendimiento del algoritmo en instancias más grandes, es decir, más cercanas a la realidad. Para lo anterior se tienen instancias de tamaño mediano presentes en el estado del arte y una instancia de real con 500 granjas.

4.2. Instancias de prueba

Para el proceso de experimentación se confeccionarán 3 conjuntos de prueba. El primero comprende un subconjunto de 22 instancias presentes en el estado del arte, siendo las que poseen una cantidad de granjas menor a 60, correspondientes a instancias pequeñas del estado del arte, mientras que el segundo grupo poseerá 15 instancias con una cantidad de

granjas comprendida entre 60 y 79. Todas estas instancias corresponden a instancias modificadas por Paredes-Belmar [Paredes-Belmar *et al.*, 2016] a partir de instancias del problema de enrutamiento de vehículos, teniendo tres camiones y tres tipos de leche, realizando una asignación de tipo de leche a cada nodo de manera cíclica, asignando al nodo 1 producción de leche de tipo A, al segundo de tipo B y al tercero C, comenzando nuevamente con leche de tipo A para el cuarto nodo. Respecto a la producción, capacidad de camiones y cuotas exigidas, estas son definidas de forma arbitraria. La leche de tipo A es la que aporta un mayor beneficio, correspondiente a 1 unidad por litro de leche entregado, mientras que la de tipo B aporta un beneficio igual a 0.7 y por último, la de tipo C posee el menor beneficio con 0.3 unidades por litro. El tercer conjunto de prueba, corresponde a una instancia real con 500 granjas, construida por los mismos autores que las instancias anteriores.

4.2.1. Instancias pequeñas del estado del arte

Las instancias pequeñas corresponden a un subconjunto de las utilizadas en [Paredes-Belmar *et al.*, 2016], estando incluidas:

- 6 instancias obtenidas del *TSPLib set* [Reinelt, 1991], identificadas por el prefijo *eil* o *att*.
- 1 instancia obtenida del trabajo de Fisher [Fisher, 1994], identificada con el prefijo *f*.
- 1 instancia obtenida del trabajo de Taillard [Taillard, 1999], identificada con el prefijo *c*.
- 14 instancias obtenidas del trabajo de Augerat [Augerat *et al.*, 1995], identificadas con el prefijo *a*.

Todos los prefijos se han mantenido según lo presentado en [Paredes-Belmar *et al.*, 2016] y el nombre de la instancia también indica el número de nodos presentes en esta, en donde cada nodo corresponde a una granja, o bien, a la planta procesadora. En la tabla 7 se presentan las instancias correspondientes a este conjunto, en donde *Net* corresponde al nombre de la instancia, $C[*; *; *]$ a la capacidad de los 3 camiones en miles de litros, $R[*; *; *]$ son los requerimientos de leche de tipo A, B y C respectivamente en miles de litros y $P[*; *; *]$ corresponde a la producción total de leche de cada tipo en miles de litros, ordenado de la misma manera que los requerimientos de la planta.

Cabe destacar que según lo trabajado por Villagrán en [Villagrán, 2019], se ha modificado la instancia *c50*, debido a que presentaba problemas de consistencia respecto a la capacidad de los camiones y la producción de las granjas, teniendo una producción de leche de 97.300 litros en total, pero una capacidad sumada de los camiones igual a 95.000 litros, siendo imposible poder obtener una solución factible. Por lo anterior, se aumenta la capacidad de 2 camiones

Tabla 7: Instancias pequeñas del estado del arte.

Fuente: Elaboración propia

<i>Net</i>	<i>Q</i>	<i>R</i>	<i>P</i>
eil22	[10;15;20]	[6;5;4]	[9.8;7.2;5.5]
eil23	[6;7;8]	[1; 1.5; 2]	[6.1;2;2]
eil30	[5;5.5;6]	[2.2;2.4;2.6]	[2.4;6.8;3.5]
eil33	[20;20;15]	[8;7;6]	[11.2;11.6;6.4]
eil51	[25;30;35]	[22;23;24]	[22.4;29.6;25.7]
att48	[50;45;40]	[38;34;30]	[40;40;37.5]
f45	[20;15;10]	[5;10;5]	[9.8;17.9;8.3]
c50	[35;35;35]	[25;20;15]	[31.1;34.5;29.7]
a33	[15;20;25]	[15;8;6]	[17.6;11.4;15.6]
a34	[20;20;25]	[10;12;14]	[15.9;16.2;13.9]
a36	[20;15;15]	[10;12;14]	[16;14.9;13.3]
a37	[20;15;10]	[10;8;6]	[13.5;12.7;14.5]
a38	[20;20;10]	[10;15;15]	[14.3;19.2;14.6]
a39	[20;20;20]	[10;12;14]	[14.6;17.3;16.6]
a44	[25;20;15]	[20;16;12]	[24.4;17;15.6]
a45	[25;20;20]	[20;18;18]	[24.3;18;17]
a46	[30;25;20]	[16;17;18]	[22.4;19.9;18]
a48	[30;25;20]	[20;20;20]	[20.6;19.5;22.5]
a53	[30;30;30]	[20;20;20]	[23.4;24.7;23.2]
a54	[15;15;15]	[5;5;5]	[12.6;11.6;9.2]
a55	[15;15;20]	[5;10;15]	[11.9;12.2;17.8]
a60	[20;10;20]	[8;12;16]	[11.8;14.6;15]

de 30.000 a 35.000, quedando los tres camiones con dicha capacidad y siendo posible transportar toda la leche. Además, las demandas presentadas en [Paredes-Belmar *et al.*, 2016] son modificadas de 15.000, 30.000 y 26.000 litros de leche de tipo A, B y C a 25.000, 20.000 y 15.000 litros respectivamente.

Según se puede ver en la tabla 7, los camiones dan abasto sin problemas para cumplir con las cuotas exigidas por la planta procesadora en caso de que se haga una asignación favorable para cada tipo de camión, pudiendo darse el caso de que, por ejemplo, para la instancia *a60* se asigne el camión con capacidad de 10 para transportar leche de tipo C cuya demanda es de 16.000, teniendo que, inevitablemente, incurrir a realizar mezcla en la planta procesadora, en caso de que se mantenga ese camión durante toda la ejecución. Respecto a la producción total de cada tipo de leche, se puede verificar que en casi todos los casos existe leche suficiente para satisfacer cada cuota, siendo la excepción, la instancia *a48*, en la que la producción total de leche de tipo B no es capaz de suplir la demanda de la planta. Si se compara la producción con las cuotas exigidas, se observa que, en la mayoría, se presenta una producción mucho mayor a lo solicitado, mientras que en unos pocos casos la diferencia

respecto a lo solicitado es pequeña o nula (como en la instancia *eil23* con la leche de tipo C).

4.2.2. Instancias medianas del estado del arte

Las instancias medianas corresponden a un subconjunto de las utilizadas por [Paredes-Belmar *et al.*, 2016], estando incluidas:

- 1 instancia obtenida del *TSPLib* set [Reinelt, 1991], identificada con el prefijo *eil*.
- 2 instancias obtenidas del trabajo de Fisher [Fisher, 1994], identificadas con el prefijo *f*.
- 5 instancias obtenidas del trabajo de Taillard [Taillard, 1999], identificadas con el prefijo *tai* o *c*.
- 7 instancias obtenidas del trabajo de Augerat [Augerat *et al.*, 1995], identificadas con el prefijo *a*.

Al igual que en las instancias pequeñas, se ocupa la misma nomenclatura de la tabla 7. Las características de estas instancias se muestran en la tabla 8.

Tabla 8: Instancias medianas del estado del arte.
Fuente: Elaboración propia

<i>Net</i>	<i>Q</i>	<i>R</i>	<i>P</i>
<i>eil76</i>	[45;48;51]	[45;43;40]	[46.8;46.7;42.9]
<i>f71</i>	[50;50;50]	[25;20;10]	[26.8;54.6;33.3]
<i>f72</i>	[60;60;60]	[20;30;40]	[26.8;54.6;33.3]
<i>tai75A</i>	[25;30;35]	[10;15;20]	[25.1;11.8;31.7]
<i>tai75B</i>	[30;30;30]	[20;25;25]	[27.1;23.9;23.4]
<i>tai75C</i>	[15;20;25]	[5;10;15]	[14.1;8.9;24.5]
<i>tai75D</i>	[20;25;35]	[15;15;20]	[20.9;19.8;30]
<i>c75</i>	[30;30;30]	[10;25;25]	[23.4;23.3;21.4]
<i>a61</i>	[35;35;35]	[30;20;10]	[30.4;34.6;23.5]
<i>a62</i>	[15;15;15]	[10;11;12]	[12.6;10.8;13.1]
<i>a63</i>	[20;20;20]	[5;10;20]	[10;17.2;16.4]
<i>a64</i>	[20;20;20]	[5;10;20]	[11.7;16;14.6]
<i>a65</i>	[15;15;15]	[10;12;14]	[14.3;15.3;14.1]
<i>a69</i>	[20;20;20]	[10;15;15]	[11.7;15.7;14.8]
<i>a80</i>	[20;20;20]	[16;10;16]	[16.2;14.6;16.2]

En este conjunto también se deben modificar dos instancia según lo expuesto por Villagrán en [Villagrán, 2019]. La primera corresponde a la instancia *c75*, debido a que existen problemas de consistencia respecto a que la producción total de las granjas no es capaz de satisfacer

las cuotas exigidas por la planta, ya que la producción total es de 68.200 litros (distinta a la presentada originalmente) distribuyéndose en 23.400 de tipo A, 23.350 de tipo B y 21.450 de tipo C, mientras que los requerimientos de la planta son 40.000, 45.000 y 50.000 respectivamente, siendo infactible poder obtener una solución que cumpliera dichos requisitos. En este caso los requerimientos se disminuyen a 10.000 para la leche de tipo A, 25.000 para la de tipo B y 25.000 para la de tipo C, pudiendo notar que será necesario realizar mezcla para poder satisfacer las cuotas de tipo B y C. También se modifica la capacidad de los camiones, pasando de tener uno con 40.000 litros y dos con 50.000 litros a tres camiones con 30.000 litros. La segunda instancia a modificar corresponde a *tai75D*, en esta sólo se modifica la capacidad de un camión en 5.000 litros, pasando de 30.000 a 25.000 litros.

En la tabla 8, se puede observar que los camiones pueden cumplir sin problemas con las cuotas exigidas con cualquier tipo de asignación. Respecto a la producción total de leche, en la mayoría de los casos, es superior a las cuotas que solicita la planta procesadora para cada tipo de leche, variando en la cantidad de leche extra que se produce. Sin embargo, en algunas instancias como *tai75A* se necesitará realizar mezcla de algún tipo para poder satisfacer las cuotas de tipo B y C, lo mismo ocurre con las instancias *tai75B*, *tai75C*, *c75* y *a62*. Por otro lado, para las instancias *tai75D*, *a64* y *a80* se necesitará realizar mezcla de algún tipo para satisfacer la cuota de tipo C.

4.2.3. Instancia Real

Esta instancia fue construida por Paredes-Belmar *et al.* a partir de la información de 500 productores distribuidos en una región geográfica de aproximadamente 9600 km², en la región de Los Lagos, Chile. La producción de estos varía entre los 57 y 25.000 litros, produciendo en su mayoría, menos de 5.000 litros, mientras que el tipo de leche que producen se presenta de la siguiente manera:

- 313 granjas producen leche de tipo A, sumando un total de 1.435.168 litros.
- 159 granjas producen leche de tipo B, sumando un total de 268.564 litros.
- 28 granjas producen leche de tipo C, sumando un total de 74475 litros.

La demanda por parte de la planta corresponde a 1.250.000 litros de leche de tipo A, 300.000 litros de tipo B y 100.000 litros de tipo C, mientras que, el beneficio aportado por litro de cada tipo de leche, no es el usado para las instancias anteriores, siendo $1,5 \cdot 10^{-2}$, $1,05 \cdot 10^{-2}$ y $4,5 \cdot 10^{-3}$ para las leches de tipo A, B y C respectivamente. Otro cambio respecto a las instancias del estado del arte es el número de camiones, siendo igual a 100 y componiéndose de la siguiente manera:

- 15 camiones de 15.000 litros

- 20 camiones de 20.000 litros
- 15 camiones de 25.000 litros
- 50 camiones de 30.000 litros

En la figura 9, se puede ver la distribución de las granjas en el área geográfica en la que se encuentran. En dicha figura los triángulos representan a los productores de leche de tipo A, los círculos a productores de tipo B, mientras que los cuadrados corresponden a granjas productoras de leche de tipo C, además, se representa a la planta con un diamante cerca del centro del mapa.



Figura 9: Distribución de las granjas en la región geográfica
Fuente: Paredes-Belmar *et al.*

La última diferencia respecto a las instancias del estado del arte, corresponde a que la matriz de distancias entre granjas no es simétrica, debido a que, al ser un caso real, la distancia que existe entre dos nodos no será necesariamente la misma. El motivo de lo anterior es que los caminos que se recorren no necesariamente serán el mismo, teniendo que considerar factores como, por ejemplo, vías o pendientes de un solo sentido, peajes, o la zona por la que

se transita (carretera, zona urbana, rural, etc). Todo lo anterior puede ser considerado para hacer variar los costos de moverse de una granja a otra o a la planta procesadora, elemento que sólo fue calculado como la distancia euclidiana entre los nodos para las instancias del estado del arte.

4.3. Entorno de experimentación

Los experimentos se realizaron en un equipo con un procesador Ryzen 5600X con frecuencia de 3.7 GHz, 16 GB de ram DDR4 con frecuencia de 3.2 GHz y un disco de estado sólido. El sistema operativo utilizado corresponde a Manjaro 21.0.

4.4. Experimentación

El proceso de experimentación se divide en cuatro etapas, cada una con un conjunto de instancias definido y realizando un análisis de los componentes del algoritmo propuesto. Primero, se presentan las pruebas realizadas en las instancias pequeñas, en las que se evalúan distintos valores para las penalizaciones presentes en la función de evaluación, además de evidenciar qué configuración inicial es la que produce mejores resultados. La segunda etapa, una vez identificados los mejores valores para las penalizaciones, se centra en las instancias medianas y cómo afecta la configuración inicial a la cantidad de evaluaciones de vecinos que se deben hacer en ambos movimientos. Luego, se realizan pruebas con la mejor configuración de parámetros para la instancia real. Finalmente, se realizan pruebas sin hacer uso del movimiento de intercambio de camiones en la instancia real, evaluando la utilidad de este.

En todos los experimentos se contrasta lo obtenido con lo presentado por Paredes-Belmar y Villagrán.

Para las siguientes pruebas se utilizarán 10 *seeds* generadas aleatoriamente. Estas se presentan a continuación:

1. 935
2. 88251
3. 49906
4. 51321
5. 11338
6. 25531
7. 4997

8. 17883

9. 68541

10. 5325

4.4.1. Instancias pequeñas

El algoritmo propuesto trabaja con dos penalizaciones, una para el incumplimiento de las cuotas exigidas por la granja y otra para el incumplimiento de la capacidad máxima de los camiones empleados en la ruta.

La primera penalización se calcula de la siguiente manera:

$$penalizacionCuota = factorCuota \cdot litrosCuota \cdot ingresosCuota$$

En esta, *factorCuota* variará para penalizar los litros faltantes de alguna cuota con un mayor o menor valor. Cabe destacar que, la penalización depende también del ingreso que aporta un litro del tipo de leche faltante, siendo mayor en caso de que falte leche de tipo A y menor si la cuota sin cumplir es de tipo C.

La segunda penalización, por otro lado, sólo depende del factor *factorCamion* y los litros que exceden la capacidad del camión en una determinada ruta, sin distinguir entre el tipo de leche que lleva esta, calculándose como:

$$penalizacionCamion = factorCamion \cdot capacidadExcedida$$

Al realizar los experimentos se variarán los valores de ambos factores, tomando 2 valores distintos, 10 y 50. Estos valores son seleccionados de tal manera que el incumplimiento de dichas cuotas sea relevante al momento de evaluar la solución, sin embargo, en comparación al trabajo presentado en [Villagrán, 2019], se mantiene el valor de penalización que presenta los mejores resultados, correspondiente a 10, pero se decide tomar un valor menor del factor más duro usado en sus experimentos, que corresponde a 100. Lo anterior tiene el objetivo de determinar si es que un factor de penalización mayor a 10 puede llevar a mejores resultados. Con esto se generan 4 combinaciones de parámetros de penalizaciones.

Por otro lado, al crear la solución inicial, el algoritmo genera seis soluciones idénticas, que varían en el tipo mínimo de leche que puede recibir una ruta. Lo anterior es relevante al momento de realizar el primer movimiento, ya que se ignorarán vecinos que provoquen una degradación de la leche de todo el camión. Por ejemplo, una granja que produce leche de

tipo B no podrá ser ubicada en una ruta cuyo tipo mínimo sea A, pero sí a una que tenga tipo mínimo B o C.

Con esto, dado que se tienen 10 *seeds*, 4 combinaciones distintas para los parámetros de penalización y 6 soluciones iniciales, para una instancia en concreto se realizarán 240 experimentos.

En la tabla 9, se presentan los parámetros que recibe el algoritmo propuesto y los valores que se usan al momento de ser ejecutado. La primera columna define la cantidad de instancias a utilizar (correspondientes a las presentadas en la sección 4.2.1), mientras que la segunda, presenta el número máximo de iteraciones, que en este caso no es utilizado, ya que se utiliza como criterio de parada el tiempo límite. En la tercera columna, se presenta el valor en segundos del tiempo máximo de ejecución, se asignó un valor de 1000 segundos, ya que se espera que se encuentre un óptimo local antes de este tiempo. La cuarta columna define que se utilizarán las 10 *seeds* presentadas anteriormente, mientras que las columnas Pen.cuota y Pen.camión muestran los valores que toman los factores de penalización. La última columna muestra las configuraciones iniciales a utilizar, siendo en este caso 6.

La configuración inicial es representada por tres letras que indican la capacidad mínima que puede recibir un camión determinado, por lo que una configuración inicial “ABC”, asignada a una solución inicial construida, implica que el camión que lleva leche de tipo A, recibe leche sólo de tipo A, mientras que el que lleva leche de tipo B recibe leche de tipo A o B, siendo el camión que lleva leche de tipo C, el único que puede recibir de cualquier tipo, mientras que una configuración inicial “BCC” indicaría que el camión que lleva leche de tipo A puede recibir como mínimo leche de tipo B, mientras que los otros dos camiones podrían recibir cualquier tipo de leche.

Tabla 9: Parámetros para la ejecución del algoritmo y valores utilizados para instancias pequeñas.

Fuente: Elaboración propia

Instancias	Iteraciones	Tiempo límite	Seeds	Pen. cuota	Pen. camión	Cfg. inicial
22 instancias pequeñas	Ilimitadas	1000	10 <i>seeds</i>	10	10	ABC, ACC, BBC, BCC, CBC, CCC
22 instancias pequeñas	Ilimitadas	1000	10 <i>seeds</i>	10	50	ABC, ACC, BBC, BCC, CBC, CCC
22 instancias pequeñas	Ilimitadas	1000	10 <i>seeds</i>	50	10	ABC, ACC, BBC, BCC, CBC, CCC
22 instancias pequeñas	Ilimitadas	1000	10 <i>seeds</i>	50	50	ABC, ACC, BBC, BCC, CBC, CCC

La tabla 10 muestra los mejores resultados obtenidos al ejecutar el algoritmo propuesto con el subconjunto de instancias pequeñas para cada combinación. La primera columna co-

responde a la instancia a probar, mientras que las cuatro columnas siguientes presentan los resultados para cada combinación de parámetros a probar, siendo estos 10-10, 10-50, 50-10, 50-50, correspondiendo el primer número al factor de penalización para el incumplimiento de las cuotas exigidas por la planta, mientras el factor de penalización por exceder la capacidad máxima de los camiones es el segundo. Además, para cada instancia se destaca en negrita el mejor resultado obtenido, priorizando, en primera instancia, la calidad obtenida y en el caso de que más de alguna configuración obtuviese el mismo resultado, destacando la con menor tiempo de ejecución.

En cada columna se muestra la configuración inicial (Cfg) que obtuvo el mejor resultado, representada por 3 letras que indican la capacidad mínima que puede recibir cada camión. Luego, se muestra la mejor calidad obtenida para dicha instancia, la que está truncada a la unidad, debido a que las distancias entre las granjas (o la planta) pueden no ser números enteros, lo que también ocurre con el beneficio que aporta cada ruta, ya que los tipos de leche B y C aportan un beneficio de 0.7 y 0.3 respectivamente. Por último, se muestra el tiempo en segundos que tomó llegar al óptimo local, truncado a la milésima.

Tabla 10: Resultados obtenidos con distintos parámetros de penalización para instancias pequeñas.

Fuente: Elaboración propia

	10 - 10			10 - 50			50 - 10			50 - 50		
	Cfg	Calidad	Tiempo	Cfg	Calidad	Tiempo	Cfg	Calidad	Tiempo	Cfg	Calidad	Tiempo
a33	ABC	29417	0.006	ABC	29417	0.006	ABC	29417	0.006	ABC	29417	0.006
a34	CBC	30492	0.009	CBC	30492	0.009	CBC	30492	0.009	CBC	30492	0.009
a36	BBC	29230	0.011	BBC	29230	0.011	BBC	29230	0.011	BBC	29230	0.011
a37	CBC	24718	0.021	CBC	24718	0.021	BBC	24807	0.012	CBC	24718	0.021
a38	BCC	28416	0.016	CCC	28478	0.019	CBC	28492	0.025	BCC	28416	0.016
a39	BBC	30801	0.009	BBC	30801	0.009	BBC	30801	0.009	BBC	30801	0.011
a44	ACC	38643	0.02	ACC	38643	0.019	ACC	38643	0.019	ACC	38643	0.019
a45	ABC	40263	0.025	CCC	40277	0.022	BBC	40267	0.024	ABC	40263	0.026
a46	BCC	40679	0.016	BCC	40679	0.016	BCC	40679	0.015	BCC	40679	0.016
a48	ACC	39800	0.028	ACC	39800	0.028	BBC	39800	0.034	ACC	39800	0.028
a53	ABC	46646	0.028	ABC	46646	0.028	ABC	46646	0.028	ABC	46646	0.027
a54	ACC	22411	0.027	ACC	22411	0.029	ACC	22411	0.027	ACC	22411	0.029
a55	ACC	24682	0.038	ACC	24682	0.038	ACC	24682	0.038	ACC	24682	0.038
a60	BCC	24510	0.075	BCC	24510	0.074	BCC	24510	0.074	BCC	24510	0.076
att48	ABC	17447	0.028	ABC	17447	0.027	ABC	17447	0.028	ABC	17447	0.027
c50	BBC	65465	0.039	BBC	65465	0.039	BBC	65465	0.039	BBC	65465	0.039
eil22	BCC	15947	0.001	BCC	15947	0.001	BCC	15947	0.001	ABC	15947	0.001
eil23	CBC	7207	0.001	CBC	7207	0.001	CBC	7207	0.001	CBC	7207	0.001
eil30	BCC	7101	0.009	ACC	7102	0.008	BCC	7101	0.009	BCC	7101	0.008
eil33	CBC	20407	0.005	CBC	20407	0.005	CBC	20407	0.005	CBC	20407	0.005
eil51	BBC	50126	0.022	BBC	50125	0.041	CBC	50120	0.032	BBC	50126	0.023
f45	BBC	23695	0.031	BBC	23695	0.023	BBC	23695	0.024	BBC	23695	0.023

En la tabla 11 se realiza una comparación con los resultados presentados en el estado del arte, presentando primero el nombre de la instancia, seguido de la calidad y el tiempo presentado en el estado del arte. A su lado, se presenta si esta solución presenta mezcla de

leche, diferenciando si esta se realizó en la ruta de algún o algunos camiones, o bien, en la planta procesadora. En el caso de que no se haya reportado si la solución presenta mezcla de algún tipo se representará por el símbolo “-”.

A continuación, se presenta la configuración inicial del mejor resultado y el GAP de esta al óptimo presentado en la literatura.

El GAP corresponde al error de la solución obtenida, con respecto al mejor valor conocido presentado en la literatura, siendo este positivo cuando la solución obtenida posee una calidad menor que la publicada, cero cuando la solución coincide con el óptimo presentado y tomando un valor negativo en caso de que se supere el valor del óptimo. Se calcula de la siguiente manera:

$$GAP = MejorValorLiteratura - MejorValorObtenido$$

Se decide utilizar esta forma de calcular el GAP, similar al error absoluto pero distinguiendo si la diferencia producida es debido a si el valor obtenido es menor (tomando un valor positivo) o mayor (tomando un valor negativo). Se descarta utilizar el error relativo ya que en la mayoría de los casos es menor a un 1 %.

Finalmente, al igual que en la solución reportada en la literatura, se muestra si la mejor solución presenta mezcla de leche, distinguiendo si esta se realizó en la ruta de algún o algunos camiones o en la planta procesadora.

Respecto a los resultados obtenidos, la mayoría de las instancias consiguen llegar al mismo óptimo local, todos factibles, independiente de cuál sea la combinación de parámetros de penalización aplicada, siendo la excepción las pruebas realizadas con las instancias *a37*, *a38*, *a45*, *eil30* y *eil51*, en las que se presentan dos o tres óptimos distintos. Para todas las instancias se obtienen tiempos de ejecución menores a un segundo, presentando una tendencia a crecer a medida que se trabaja con instancias con un mayor número de granjas, lo que es esperable. Por otro lado, los mejores resultados, siendo elegidos generalmente por un menor tiempo de ejecución, son escogidos por diferencias de valores menores a la milésima, lo que puede llegar a ser, de cierta manera, variable, al tener que medir ventanas de tiempo tan pequeñas.

La mejor combinación de parámetros, corresponde a la que posee un factor de penalización por incumplimiento de cuotas para las plantas igual a 50 y un factor de penalización de 10 para el exceso de capacidad del camión de alguna ruta. Esto es tomado en cuenta, principalmente, por las pequeñas diferencias en el tiempo de ejecución, que producen que tenga un mayor número de mejores resultados. Además, esta combinación obtiene óptimos mayores que las otras combinaciones para dos instancias (*a37* y *a38*). La combinación 10-50 es otra que obtiene óptimos mayores para otras dos instancias (*a45* y *eil30*). Un motivo para seleccionar una mayor penalización al incumplimiento de las cuotas que a la sobrecarga de

Tabla 11: Mejores resultados obtenidos para instancias pequeñas.

Fuente: Elaboración propia

	Estado del arte		Mezcla		Mejor resultado		Mezcla	
	Calidad	Tiempo	En ruta	En planta	Cfg	GAP	En ruta	En planta
a33	29417	62	no	no	ABC	0	no	no
a34	30496	40	no	sí	CBC	4	no	sí
a36	29233	110	no	sí	BBC	3	no	sí
a37	24837	45	sí	no	BBC	30	sí	no
a38	28596	570	sí	no	CBC	104	sí	no
a39	30808	110	no	no	BBC	7	no	no
a44	38771	101	sí	sí	ACC	128	sí	sí
a45	40282	136	sí	no	CCC	5	sí	no
a46	40696	66	no	no	BCC	17	no	no
a48	39800	230	sí	sí	ACC	0	sí	sí
a53	46662	183	no	no	ABC	16	no	no
a54	22414	304	no	no	ACC	3	no	no
a55	24694	270	no	no	ACC	12	no	no
a60	24587	3565	sí	sí	BCC	77	sí	sí
att48	17452	284	sí	no	ABC	5	sí	no
c50	65477	84	-	-	BBC	12	no	no
eil22	15947	12	no	no	ABC	0	no	no
eil23	7207	6	no	no	CBC	0	no	no
eil30	7117	99	sí	no	ACC	15	sí	no
eil33	20409	58	no	no	CBC	2	no	no
eil51	50128	154	no	no	BBC	2	no	no
f45	23705	80	sí	no	BBC	10	sí	no

los camiones se basa en que, en las primeras iteraciones, en donde la calidad de la solución puede ser baja (o incluso negativa), si se realiza la reubicación de una granja a otra ruta cuya calidad mínima es mayor y esto significa en un aumento de la calidad de la solución, a medida que avancen las iteraciones, le será más difícil al algoritmo reubicar dichas granjas que empeoran una ruta entera y provocan que, de plano, no se pueda cumplir con la cuota de algún tipo de leche, en contraste con preferir exceder la capacidad del camión, ya que el algoritmo puede solventar esto con el movimiento de intercambio de camiones o bien con el movimiento de reubicación de granja. Por esta situación, en algunas pruebas realizadas, se obtuvo un óptimo local con una calidad negativa, debido a las penalizaciones incurridas por no poder cumplir con las cuotas de la planta.

La configuración inicial es bastante diversa, pero presenta una tendencia a permitir evaluar vecinos que muevan granjas de rutas que poseen una mayor calidad, presentando en su gran mayoría, rutas con configuración inicial ACC, BBC, BCC y CBC, seguida de la configuración más restrictiva, ABC, para ciertas instancias y siendo la configuración CCC la que menos veces se

hace presente, siendo esta elegida sólo 2 veces, lo que se puede deber principalmente a que, dado que debe construir y evaluar un mayor número de vecinos, el tiempo de ejecución es mayor.

Pasando a la comparación de los mejores resultados obtenidos con los presentados en el estado del arte en la tabla 11, se debe ratificar que las calidades y tiempos presentados son los correspondientes a Paredes-Belmar, exceptuando dos instancias, *a60* y *c50*. Esto se debe, a que en el trabajo de Villagrán, se evidenció que la instancia *c50* presentaba problemas de consistencia, modificando la capacidad de los camiones y las demandas de la planta, mientras que se evalúa que la calidad del óptimo de la instancia *a60* sea menor, siendo imposible obtener resultados que superasen una calidad de 24587. De las 22 instancias probadas, sólo se pudo obtener el óptimo del estado del arte en 4 de ellas, mientras que en el resto de estas, el GAP presentado es pequeño en comparación a la magnitud de la calidad, teniendo una diferencia menor al 1 % en todos los casos. Además, se sospecha que, en las instancias cuyo GAP es menor a 5 unidades, este esté dado debido a que en ningún momento se redondea la calidad de la solución, sólo se trunca el resultado a la unidad, lo que podría cambiar si se hace un redondeo a la calidad de cada una de las rutas y/o bien a la calidad de la solución en sí.

Finalmente, se evidencia que la mezcla de leche es relevante, siendo presente ya sea, transportando leche de más de un tipo en un camión y/o realizando una mezcla luego de “entregar” la leche a la planta, en el óptimo local encontrado en 11 instancias. Se destaca además, que se presentan los mismos tipos de mezcla para todas las instancias, excepto para la instancia *c50*, que fue modificada por Villagrán, ya que no se reporta si existió mezcla de algún tipo. En el trabajo de Paredes-Belmar se presenta mezcla de ambos tipos, lo que se puede deber a la falta de capacidad de los camiones.

4.4.2. Instancias medianas

Para las instancias medianas se estudia cómo se comportan los movimientos propuestos, la reubicación de granjas y el intercambio de camiones, en base a las distintas configuraciones iniciales planteadas anteriormente. Esta vez, se fijan los parámetros para los factores de penalización a 50 y 10, para el de incumplimiento de cuotas y la sobrecarga de camiones respectivamente.

Se reutilizarán las 10 *seeds* presentadas anteriormente, probando 6 configuraciones iniciales, generando así 60 experimentos para una instancia en concreto. En la tabla 12 se presentan los parámetros que recibe el algoritmo propuesto y los valores que se usan al momento de ser ejecutado. La primera columna define la cantidad de instancias a utilizar (correspondientes a las presentadas en la sección 4.2.2), mientras que la segunda presenta el número máximo de iteraciones, que se presenta como ilimitado, ya que se utilizará como criterio de parada el tiempo límite. En la tercera columna, se presenta el valor en segundos del tiempo máximo de ejecución, que al igual que para las instancias pequeñas, será de 1000 segundos.

La cuarta columna define la cantidad de *seeds* que se utilizarán, que nuevamente corresponden a las presentadas anteriormente. Luego, se presentan las columnas correspondientes a los factores de penalización. La última columna presenta las configuraciones iniciales a utilizar, manteniendo las 6 posibilidades.

Tabla 12: Parámetros para la ejecución del algoritmo y valores utilizados para instancias medianas.

Fuente: Elaboración propia

Instancias	Iteraciones	Tiempo límite	Seeds	Pen. cuota	Pen. camión	Cfg. inicial
15 instancias medianas	Ilimitadas	1000	10 seeds	50	10	ABC, ACC, BBC, BCC, CBC, CCC

Las tablas 13, 14 y 15 muestran, para cada instancia, el mejor resultado obtenido al ejecutar el algoritmo propuesto para cada configuración inicial, junto con el promedio de las calidades obtenidas, el promedio de vecinos construidos y evaluados para el movimiento de reubicación de granja (Prom. mov. 1) y para el movimiento de intercambio de camiones (Prom. mov. 2), además de presentar el promedio de tiempo obtenido de cada configuración inicial obtenido con las 10 *seeds*. Los tres primeros valores se encuentran truncados a la unidad, mientras que el promedio del tiempo se encuentra truncado a la milésima. Se destaca en negrita el mejor resultado obtenido para cada instancia y el mejor promedio de calidades.

En la tabla 16 se realiza una comparación con los resultados presentados en el estado del arte, mostrando primero el nombre de la instancia, seguido de la calidad y el tiempo reportado en el estado del arte. A su lado se presenta si esta solución utiliza mezcla de leche, diferenciando si esta se realizó en la ruta de algún o algunos camiones, o bien, en la planta procesadora. En caso de que no se haya reportado si la solución presenta mezcla de algún tipo, se representará por el símbolo “-”.

Luego se presenta la configuración inicial del mejor resultado y el GAP de esta al mejor valor conocido reportado en la literatura. Finalmente, se muestra si la mejor solución presenta mezcla de leche, distinguiendo si esta se realizó en la ruta de algún o algunos camiones o en la planta procesadora.

Al fijar los factores de penalización, se pueden notar diferencias entre los mejores resultados obtenidos para cada configuración inicial, coincidiendo en el mismo óptimo local sólo una vez en la instancia *tai75A*, destacando, además, que todos estos corresponden a soluciones factibles. La mayoría de los mejores resultados y mejores promedios de las calidades obtenidas, se presentan en configuraciones iniciales que permiten crear y evaluar vecinos del movimiento de reubicación de granja, que lleven a una granja de menor calidad a una ruta de mayor calidad, pudiendo ser esta tanto la ruta encargada de la leche de tipo A como la de tipo B. Las configuraciones que obtienen el mejor resultado y el mejor promedio un menor número de veces corresponden a la más restrictiva, ABC y la menos restrictiva CCC, mientras que la más destacada corresponde a la configuración BCC. Lo anterior, puede estar dado por el hecho de que, para cada ruta, sólo permite probar con vecinos que muevan una

Tabla 13: Resultados obtenidos para distintas configuraciones iniciales de instancias medianas.

Fuente: Elaboración propia

		ABC	ACC	BBC	BCC	CBC	CCC
a61	Mejor resultado	60576	60623	60554	60599	60601	60564
	Prom. calidad	60503	60500	60481	60531	60487	60472
	Prom. mov. 1	11222	14529	14500	18372	16559	19427
	Prom. mov. 2	3729	7596	4302	8612	4274	8053
	Promedio tiempo	0.043	0.055	0.049	0.064	0.052	0.065
a62	Mejor resultado	22819	22835	22820	22836	22785	22854
	Prom. calidad	22733	22731	22728	22745	22737	22740
	Prom. mov. 1	13086	17577	16702	18765	20399	22472
	Prom. mov. 2	0	0	0	0	0	0
	Promedio tiempo	0.046	0.053	0.052	0.055	0.059	0.063
a63	Mejor resultado	24376	24397	24415	24364	24427	24384
	Prom. calidad	24164	24142	24175	24123	24178	24134
	Prom. mov. 1	15814	18306	20231	18409	22753	20492
	Prom. mov. 2	7929	7793	8484	7192	8818	6775
	Promedio tiempo	0.066	0.07	0.075	0.068	0.08	0.071
a64	Mejor resultado	24022	24050	24060	24024	24006	24040
	Prom. calidad	23961	23964	23946	23921	23943	23955
	Prom. mov. 1	17872	19219	18434	19220	21971	24833
	Prom. mov. 2	9996	8331	8546	8821	8918	10122
	Promedio tiempo	0.076	0.076	0.072	0.074	0.08	0.087
a65	Mejor resultado	27939	27994	27836	27822	27848	27968
	Prom. calidad	26880	27202	27350	27752	27602	27802
	Prom. mov. 1	14820	19583	20961	18997	21706	24267
	Prom. mov. 2	14446	20392	20738	19300	21893	24396
	Promedio tiempo	0.072	0.091	0.093	0.087	0.095	0.105

granja a una ruta de su misma calidad, o bien, a la ruta que transporta el siguiente mejor tipo en términos de calidad, realizando una exploración del espacio de búsqueda más amplia, pero evitando evaluar vecinos que empeoren la leche de tipo A a C, lo que provocaría una reducción de los beneficios y una penalización significativas.

En las tablas 13, 14 y 15, se observa una tendencia a que los promedios de vecinos evaluados, para el movimiento de reubicación de granja, crecen a medida que se permite mover granjas a rutas de mejor calidad, ocurriendo lo mismo para el tiempo de ejecución. Es esperable que la configuración inicial afecte la cantidad de vecinos que se construyen y evalúan, ya que se debe probar con una mayor cantidad de reubicaciones antes de obtener alguna mejora en la calidad de la solución.

Por otro lado, existen instancias en las que no se realizó ningún movimiento de intercambio

Tabla 14: Resultados obtenidos para distintas configuraciones iniciales de instancias medianas.

Fuente: Elaboración propia

		ABC	ACC	BBC	BCC	CBC	CCC
a69	Mejor resultado	25787	25775	25783	25742	25787	25769
	Prom. calidad	25717	25687	25700	25706	25693	25697
	Prom. mov. 1	17022	20259	18653	23718	22305	30205
	Prom. mov. 2	0	0	0	0	0	0
	Promedio tiempo	0.06	0.065	0.061	0.072	0.068	0.084
a80	Mejor resultado	29909	29903	29900	29934	29838	29864
	Prom. calidad	29810	29816	29800	29771	29774	29789
	Prom. mov. 1	22245	31910	29039	42637	33838	38018
	Prom. mov. 2	0	0	0	0	0	0
	Promedio tiempo	0.087	0.107	0.099	0.129	0.108	0.117
c75	Mejor resultado	42342	42315	42338	42349	42340	42341
	Prom. calidad	42305	42281	42280	42308	42290	42282
	Prom. mov. 1	27832	34059	28241	32183	36600	32998
	Prom. mov. 2	0	0	0	0	0	0
	Promedio tiempo	0.102	0.113	0.098	0.107	0.118	0.108
eil76	Mejor resultado	91404	91392	91419	91418	91436	91367
	Prom. calidad	91376	91362	91349	91382	91377	91344
	Prom. mov. 1	20455	28947	26446	29827	27507	34359
	Prom. mov. 2	4661	6185	8673	9776	10199	12473
	Promedio tiempo	0.085	0.107	0.104	0.113	0.109	0.128
f71	Mejor resultado	72771	72754	72817	72758	72782	72750
	Prom. calidad	72442	72644	72457	72128	72373	72141
	Prom. mov. 1	22095	23712	23867	33151	27454	28027
	Prom. mov. 2	4726	9559	4194	5877	4359	4983
	Promedio tiempo	0.084	0.093	0.086	0.107	0.092	0.094

de camiones, lo que está dado por la capacidad de los camiones y la producción de cada tipo de leche, teniendo cada camión la capacidad suficiente para llevar la totalidad de la producción de cualquier tipo de leche y la capacidad restante suficiente para evaluar reubicaciones de granjas a estas, sin sobrecargar el camión. En el resto de las instancias, este movimiento es utilizado una gran cantidad de veces y no presenta la misma tendencia que el movimiento de reubicación de granja, respecto a las configuraciones iniciales.

Respecto a la comparación de los mejores resultados obtenidos con los presentados en el estado del arte, que se muestran en la tabla 16, se debe dar cuenta que las calidades y tiempos presentados son los correspondientes a Paredes-Belmar, exceptuando 3 instancias, c75, tai75A y tai75D. Lo anterior se debe a que en el trabajo de Villagrán se realizaron modificaciones a los requerimientos de la planta para la instancia c75, ya que presentaba problemas de consistencia, no siendo capaz de suplir los requerimientos originales de la planta de

Tabla 15: Resultados obtenidos para distintas configuraciones iniciales de instancias medianas.

Fuente: Elaboración propia

		ABC	ACC	BBC	BCC	CBC	CCC
f72	Mejor resultado	72030	72020	72028	72024	72034	72034
	Prom. calidad	72005	71996	72000	72002	71993	72005
	Prom. mov. 1	23653	25095	29568	30019	32651	30385
	Prom. mov. 2	410	426	472	454	427	379
	Promedio tiempo	0.087	0.088	0.098	0.099	0.103	0.097
tai75A	Mejor resultado	40292	40377	40372	40377	40389	40366
	Prom. calidad	40246	39453	40284	39440	40261	37874
	Prom. mov. 1	24194	35480	27784	34040	30350	36974
	Prom. mov. 2	660	724	754	719	793	894
	Promedio tiempo	0.092	0.119	0.102	0.112	0.106	0.122
tai75B	Mejor resultado	48187	48215	48148	48219	48198	48150
	Prom. calidad	48058	48058	48063	48114	48129	48052
	Prom. mov. 1	28158	37709	34536	37024	35165	41196
	Prom. mov. 2	642	763	765	784	743	754
	Promedio tiempo	0.113	0.132	0.124	0.129	0.125	0.136
tai75C	Mejor resultado	25846	25841	25868	25828	25850	25833
	Prom. calidad	25716	25780	25759	22033	25760	25728
	Prom. mov. 1	26633	32840	30210	37210	33836	38549
	Prom. mov. 2	4793	4942	4735	4927	5137	5528
	Promedio tiempo	0.108	0.12	0.113	0.128	0.121	0.13
tai75D	Mejor resultado	42380	42384	42419	42367	42398	42410
	Prom. calidad	42217	42254	42192	42168	42259	42186
	Prom. mov. 1	21115	23004	27359	29364	29926	32644
	Prom. mov. 2	4226	8325	5287	9376	5069	9630
	Promedio tiempo	0.09	0.1	0.105	0.114	0.108	0.122

ninguna manera. En cuanto a la instancia *tai75A*, el beneficio máximo que se puede obtener (sin contar costos ni mezcla) corresponde a 42992, mientras que el óptimo publicado por Paredes-Belmar es de 65477, siendo imposible de obtener, por lo que se considerará el publicado por Villagrán. Por último, la instancia *tai75D*, presenta la misma situación que la anterior, en la que no se puede obtener el óptimo presentado.

En las 15 instancias probadas no se pudo obtener el óptimo global, sin embargo, la diferencia del resultado obtenido al óptimo es menor al 1 % en todos los casos. Además, se debe recalcar que todas las instancias terminan su ejecución en un tiempo menor a un segundo.

Nuevamente, muchas de las instancias trabajadas hacen uso de la mezcla de leche, presentándose en el óptimo local de 11 instancias, ya sea transportando leche de distintas calidades en un mismo camión o mezclando la leche en la planta procesadora. En estas se coincide con

Tabla 16: Mejores resultados obtenidos para instancias medianas.

Fuente: Elaboración propia

	Estado del arte		Mezcla		Mejor resultado		Mezcla	
	Calidad	Tiempo	En ruta	En planta	Cfg	GAP	En ruta	En planta
a61	60644	561	no	no	ACC	21	no	no
a62	22917	1022	sí	no	CCC	63	sí	no
a63	24447	2930	sí	sí	CBC	20	sí	sí
a64	24100	5395	sí	sí	BBC	40	sí	sí
a65	28046	478	sí	no	ACC	52	sí	no
a69	25822	1552	sí	no	ABC	35	sí	no
a80	29977	5626	no	no	BCC	43	no	no
c75	46180	13760	-	-	BCC	3831	sí	sí
eil76	91461	1700	no	no	CBC	25	no	no
f71	72864	3483	sí	no	BBC	47	sí	no
f72	72072	3483	sí	sí	CCC	38	sí	sí
tai75A	42992	4806	-	-	CBC	2603	sí	sí
tai75B	48238	15056	sí	sí	BCC	19	sí	sí
tai75C	25906	4537	sí	sí	BBC	38	sí	sí
tai75D	43869	2645	-	-	BBC	1450	no	no

Paredes-Belmar respecto a los tipos de mezcla realizados en cada instancia.

Respecto a las instancias modificadas por Villagrán, *c75*, *tai75A* y *tai75D*, no se reporta si existió mezcla de algún tipo, pero el resultado obtenido con los experimentos coincide con los tipos de mezcla presentados por Paredes-Belmar. Esto se puede deber a que, a diferencia de la instancia *c50*, sólo se modifican los requerimientos por parte de la planta procesadora de la instancia *c75*, de tal manera que pueda cumplir con estos, sin modificar, por ejemplo, las capacidades de los camiones para que puedan transportar más leche, para evitar la mezcla en ruta. Mientras que, para las otras dos, sólo se verifica que no se puede obtener el óptimo reportado por Paredes-Belmar, manteniendo todos los elementos de dichas instancias sin modificar.

4.4.3. Instancia Real

Para la instancia real, construida por Paredes-Belmar, a partir de la información de 500 granjas ubicadas en la región de los Lagos en Chile, se utilizarán los mismos factores de penalización ocupados en los experimentos con instancias de mediano tamaño y además, se ocupará la configuración inicial que obtuvo mejores resultados, considerando el promedio de las calidades y la mejor calidad obtenida para cada instancia, siendo esta BCC.

Se continúa utilizando las 10 *seeds* para obtener resultados a partir de 10 soluciones iniciales

distintas. Además, dado que, a pesar de disponer de 100 camiones, se podría llegar a utilizar sólo una parte de estos y, de hecho, las soluciones iniciales no ocupan más de 80 camiones en ningún caso, se confeccionan tres instancias adicionales. Estas se diferencian en el número de camiones disponibles, siendo estos 90, 80 y 70. En cada uno se descartan los camiones con una menor capacidad. Con esto se tienen 40 pruebas en total.

La tabla 17 presenta los parámetros que recibe el algoritmo propuesto y los valores que se utilizan al momento de ser ejecutado. En la primera columna se identifica la instancia a trabajar, diferenciadas por el número de camiones disponibles en cada una de estas. La segunda columna, correspondiente al número máximo de iteraciones se presenta como ilimitado, ya que se utilizará como criterio de parada el tiempo límite. La tercera columna presenta este último parámetro en segundos, siendo 5959 para las cuatro instancias, tomando como referencia el utilizado por Paredes-Belmar al realizar experimentos sin particiones de terreno. La cuarta columna define la cantidad de *seeds* a utilizar, correspondiendo a las 10 presentadas anteriormente. Las siguientes columnas corresponden a los valores para los factores de penalización y por último, se presenta la configuración inicial utilizada, siendo esta vez, sólo una para todas las instancias.

Tabla 17: Parámetros para la ejecución del algoritmo y valores utilizados para instancia real.
Fuente: Elaboración propia

Instancia	Iteraciones	Tiempo límite	Seeds	Pen. cuotas	Pen. camión	Cfg. inicial
100 camiones (original)	Ilimitadas	5959	10 seeds	50	10	BCC
90 camiones	Ilimitadas	5959	10 seeds	50	10	BCC
80 camiones	Ilimitadas	5959	10 seeds	50	10	BCC
70 camiones	Ilimitadas	5959	10 seeds	50	10	BCC

En la tabla 18 se presenta el promedio de la calidad y tiempo obtenidos a partir de los 10 experimentos para cada número de camiones, junto con el mejor resultado obtenido y el tiempo que tomó. Todos los valores se encuentran truncados a la unidad. En la tabla 19 se muestran los mejores resultados obtenidos en la Literatura y el mejor resultado obtenido por el algoritmo propuesto, además de la utilización de mezcla de leche, tanto en ruta como en planta, por cada una de las soluciones presentadas. En el resultado de Villagrán no se informa si se utilizó mezcla de algún tipo, pero se asume que se realizó, ya que, por ejemplo, las granjas productoras de leche de tipo C no son suficientes para satisfacer la demanda de la planta procesadora. En este caso se presenta el símbolo “-” para ambos tipos de mezcla.

En todas las pruebas realizadas se llega a un óptimo local en menos de 1500 segundos, a pesar de que se realizaron pruebas con un tiempo límite de 5959 segundos. Sin embargo, en promedio, se obtiene una calidad menor a las presentadas en la literatura.

Para esta instancia, se realizan más movimientos de intercambio de camiones que de reubicaciones de granja, situación que no ocurría en las instancias pequeñas y medianas por el reducido número de camiones que utilizaban. Por otro lado, se obtienen mejores resultados al disponer de un mayor número de camiones, siendo la mejor solución la que tiene 100

Tabla 18: Resultados obtenidos con distintos número de camiones para instancia real.

Fuente: Elaboración propia

Camiones	Promedio		Mejor resultado	
	Calidad	Tiempo	Calidad	Tiempo
100	13868	899	14436	791
90	13705	770	14275	970
80	13755	802	14108	879
70	13404	688	13980	760

camiones a su disposición. Esto está dado por la posibilidad de generar rutas más pequeñas que sean menos costosas.

Tabla 19: Mejor resultado obtenido para instancia real.

Fuente: Elaboración propia

Paredes-Belmar		Villagrán		Mejor resultado	
Calidad	Tiempo	Calidad	Tiempo	Calidad	Tiempo
14155	57962	14846	899	14436	791
Mezcla en ruta	Mezcla en planta	Mezcla en ruta	Mezcla en planta	Mezcla en ruta	Mezcla en planta
sí	sí	-	-	sí	sí

Respecto al mejor resultado obtenido, se encuentra un óptimo local mayor al presentado por Paredes-Belmar, pero menor al presentado por Villagrán en, aproximadamente, un 3 %. Sin embargo, se utiliza una menor cantidad de tiempo en obtener este resultado. Además, la solución obtenida utiliza mezcla, tanto en la ruta de algún camión como en la planta procesadora y no utiliza todos los camiones que tiene disponible, al igual que ocurre con la solución presentada por Paredes-Belmar.

En la figura 10 se presenta un *boxplot* con las calidades obtenidas para cada set de pruebas de las instancias. Este corresponde a un método estandarizado para representar gráficamente una serie de datos numéricos a través de sus cuartiles. En este, se puede apreciar a simple vista la mediana (representada por una "x") y los cuartiles de los datos, además de representar los valores atípicos u *outliers* en caso de que existan.

En la figura 10 se puede apreciar la dispersión de los resultados obtenidos para cada instancia utilizada. Si bien, se presenta una distribución simétrica de los datos, para la instancia con 70 camiones, se presenta un rango más amplio para datos del segundo cuartil. Para la instancia de 90 camiones se presenta un *outlier*, que corresponde al mayor valor encontrado en las pruebas para esta instancia, mientras que el resto de los resultados presenta una menor dispersión, salvo por el primer cuartil cuyo rango es de los más grandes.

Por otro lado, la instancia con 80 camiones presenta un *outlier* con un valor muy por debajo del resto de calidades obtenidas. Este conjunto es el que presenta una menor dispersión de los datos, junto con la mayor simetría de estos. En cuanto a la instancia con 70 camiones, es la que presenta una mayor dispersión en el segundo cuartil y la mayor asimetría.

También se puede dar cuenta que la totalidad de las calidades obtenidas, que son consideradas en el *boxplot* para las instancias de 90, 80 y 70 camiones, presentan resultados inferiores a los obtenidos por Paredes-Belmar (sin contar outliers). Por otro lado, la instancia con 100 camiones, que corresponde a la que presenta mejores resultados, posee el cuarto cuartil con valores cercanos o superiores al resultado presentado por Paredes-Belmar. Con eso se puede dar cuenta que, el disponer de una mayor cantidad de camiones, a pesar de no utilizar su totalidad, logra generar soluciones cuya calidad es superior en comparación a instancias con un menor número de camiones.

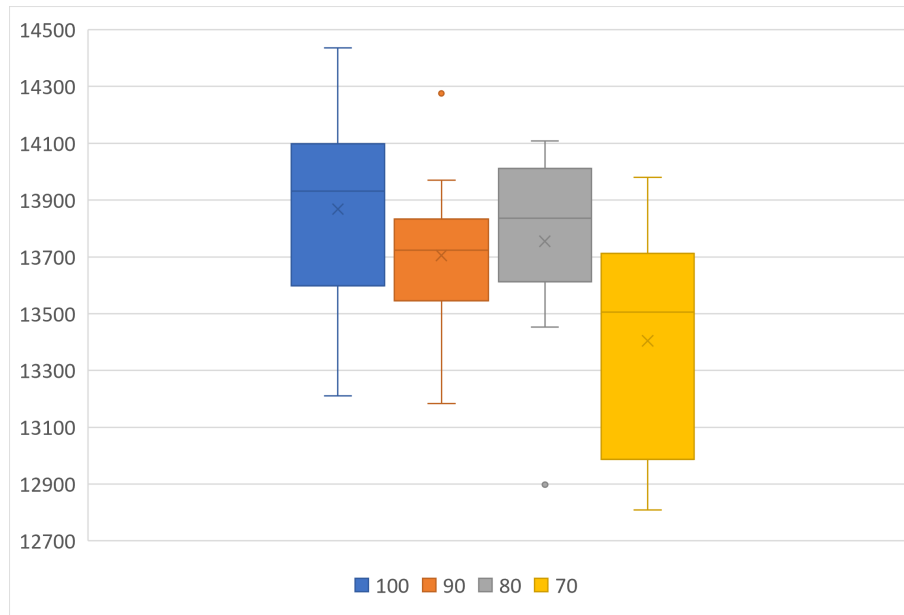


Figura 10: Boxplot en base a las calidades obtenidas por los 10 experimentos para cada número de camiones

Fuente: Elaboración propia

4.4.4. Utilidad de movimiento de intercambio de camiones

Se realizará un conjunto de pruebas con las instancias utilizadas en la sección 4.4.3 (100, 90, 80 y 70 camiones), con el objetivo de analizar el comportamiento del algoritmo cuando el movimiento de intercambio de camiones no está presente. Se pretende evidenciar la utilidad de dicho movimiento con las cuatro instancias de 500 granjas, ya que corresponden a las que poseen un número de camiones mayor a 3. De esta manera se observa la relevancia del movimiento en instancias más cercanas a la realidad.

La modificación se presenta en el algoritmo 2, en el que, luego de mover una granja, no verifica si la capacidad del camión de destino ha sido superada, por lo que se procede a comparar la calidad de la nueva solución con la actual, sin intentar reparar la sobrecarga del camión, en caso de que esta existiese.

Se mantiene el uso de las 10 *seeds* para obtener resultados a partir de 10 soluciones iniciales distintas y dado que se tienen 4 instancias con un número de camiones diferente, se tiene un total de 40 pruebas. Los valores de los parámetros utilizados son los mismos presentados en la tabla 17.

La tabla 20 presenta los resultados obtenidos. Para cada instancia a probar, identificada por el número de camiones, se presenta el promedio de la calidad y tiempo obtenidos a partir de los 10 experimentos para cada instancia, junto con el mejor resultado obtenido y el tiempo que tomó. Todos los valores se encuentran truncados a la unidad.

Tabla 20: Resultados obtenidos con distinto número de camiones para instancia real sin movimiento de intercambio de camiones.

Fuente: Elaboración propia

Camiones	Promedio		Mejor resultado	
	Calidad	Tiempo	Calidad	Tiempo
100	12508	42	12941	54
90	12760	45	13178	42
80	12849	42	13375	47
70	12770	39	13163	41

Según lo presentado en la tabla 20, se puede evidenciar un empeoramiento de la calidad de las soluciones, reduciendo el promedio de las calidades obtenidas en valores cercanos a las 1000 unidades para todas las instancias, representando una reducción de la calidad promedio de entre un 5 % a un 10 %. Lo mismo ocurre con los mejores resultados obtenidos, presentando valores menores al óptimo local reportado por Paredes-Belmar.

Otro punto a tener en cuenta, es el tiempo de ejecución, que se reduce drásticamente, siendo más de 15 veces menor. Esto implica que la inclusión del movimiento de intercambio de camiones provoca una carga de trabajo considerable, ya que cada movimiento de reubicación de granja debía realizar la tarea de verificar la capacidad del camión y ejecutar el movimiento de intercambio de camiones, en caso de ser necesario, dando la posibilidad de realizar un intercambio con un gran número de camiones antes de rechazar la reubicación de la granja.

Por otro lado, el número de iteraciones presenta valores cercanos a 1600 para la mayoría de las ejecuciones, mientras que en los experimentos realizados en la sección 4.4.3 se presenta un número de iteraciones comprendido, en su mayoría, entre 1750 y 1950.

En la figura 11 se presenta un *boxplot* con las calidades obtenidas para cada set de pruebas, mostrando además los *boxplots* obtenidos en la sección 4.4.3. En este, se muestra el resultado obtenido a partir de la ejecución que considera los dos movimientos presentados, el movimiento de reubicación de granja y de intercambio de camiones, para cada instancia en color sólido, siendo identificadas con la leyenda “*númerodecamiones* - 2 movimientos”. Al lado de cada una de estas se muestra el resultado obtenido al eliminar el movimiento de intercambio de camiones, utilizando el mismo color según el número de camiones y presentando

un relleno de franjas horizontales, siendo identificadas con la leyenda “*númerodecamiones* - 1 movimiento”.

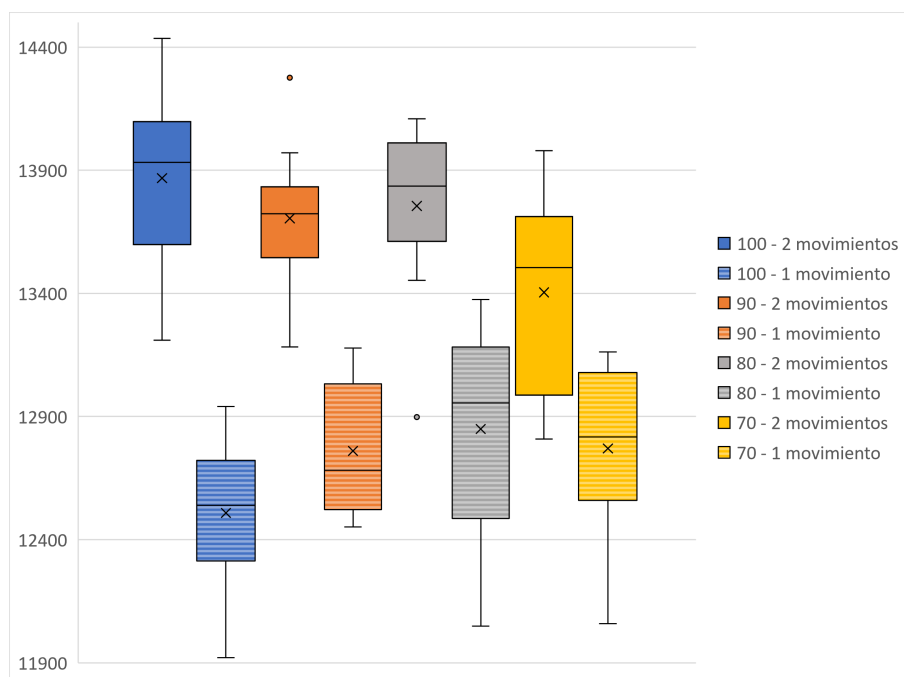


Figura 11: Boxplot de comparación de las calidades obtenidas por los 10 experimentos para cada número de camiones, con 1 y 2 movimientos

Fuente: Elaboración propia

Según lo presentado en la figura 11, se puede apreciar que los resultados obtenidos al ejecutar el algoritmo sólo con el movimiento de reubicación de granja, presentan una mayor dispersión que los obtenidos de las ejecuciones con dos movimientos. La excepción corresponde a la instancia con 70 camiones, en donde el segundo cuartil de la ejecución con dos movimientos es el que presenta una mayor dispersión, mientras que para la ejecución con un movimiento, el primer cuartil es el que presenta esta característica.

Por otro lado, se puede ver que las calidades obtenidas por las ejecuciones con un solo movimiento son menores. Esto se puede evidenciar comparando la mediana o los rangos de cada cuartil entre la ejecución con 1 movimiento y 2 movimientos, presentando valores inferiores.

Con los resultados presentados se puede evidenciar la relevancia del movimiento de intercambio de camiones en la obtención de mejores soluciones, presentando un mejor resultado, tanto en el máximo como en el promedio, si este es incluido. Se debe dar cuenta también, que la inclusión de este tiene una gran repercusión en el tiempo de ejecución.

CAPÍTULO 5

CONCLUSIONES

El problema de recolección de leche es un problema desafiante y de suma relevancia para la industria de los alimentos, dado el alto consumo de la leche por nuestra sociedad actual. No se debe olvidar que este es considerado una variante de VRP que agrega elementos que elevan su complejidad, como lo son el considerar distintos camiones, la demanda de cada tipo de leche y la posibilidad de mezcla entre los distintos tipos de esta, elementos presentes en el problema abordado en esta memoria.

Se propuso un algoritmo basado en una técnica de búsqueda local, *Hill-Climbing* con Alguna Mejora, para poder abordar el problema de recolección de la leche con mezcla, desarrollando dos movimientos que abordan distintos elementos del problema, la reubicación de las granjas y la capacidad de los camiones para poder recolectar la producción. Además, se plantearon distintas configuraciones para las soluciones iniciales, que limitaban el movimiento de granjas entre rutas, con el objetivo de no evaluar vecinos que directamente empeoren la calidad de la solución.

Para la validación de la solución se trabajó con tres subconjuntos de instancias, evaluando distintos componentes del algoritmo, de tal manera que estos sean sintonizados para poder obtener buenos resultados en la instancia real, que distaba en complejidad respecto a las presentadas en el estado del arte. Para esto, se evaluaron los valores que pueden tomar los factores de penalización, la configuración inicial escogida y cómo esta afecta a los vecinos que se construyen y evalúan durante la ejecución del algoritmo. Además, se evaluó la relevancia del movimiento de intercambio de camiones y su repercusión al momento de obtener soluciones de mejor calidad.

En cuanto a los resultados obtenidos, si bien no se pudo obtener el mejor valor reportado en la literatura, en la mayoría de las instancias, la diferencia de calidad es pequeña, siendo menor a un 1 % para las instancias del estado del arte y menor a un 3 % para la instancia real. Por otro lado, el tiempo de ejecución requerido para obtener un óptimo local, es menor al presentado en la literatura para todas las instancias, siendo menor a 1 segundo para instancias pequeñas y medianas, mientras que para la instancia real se obtuvieron tiempos menores a 1500 segundos en todas las pruebas realizadas.

Por otro lado, respecto a los parámetros de penalización, se evidenció que se obtienen mejores resultados al utilizar un factor de penalización mayor para el incumplimiento de cuotas, en comparación al factor de penalización relacionado a la sobrecarga de los camiones.

Se dio cuenta de la importancia de reparar la solución por medio del intercambio de camiones, en el caso de que, al reubicar una granja, esta exceda la capacidad del camión asignado a la ruta. Se pudo ver una reducción significativa de las calidades obtenidas al momento de excluir este movimiento.

Se destaca el hecho de que las configuraciones iniciales con mejores resultados corresponden a las que permiten un mayor grado de libertad al momento de reubicar una granja, permitiendo evaluar soluciones que podrían empeorar la calidad completa de una ruta y ser penalizadas, en comparación a movimientos más restrictivos como en la configuración ABC, en donde las rutas sólo podían recibir granjas de produjesen leche de su calidad o superior.

Como trabajo futuro sería interesante explorar la adición de otros movimientos al algoritmo, capaces de modificar las rutas de los camiones de forma más drástica, como intercambiar las posiciones entre dos granjas, o aplicar movimientos de reparaciones de rutas de manera individual para cada camión como 2-*OPT* o 3-*OPT*, considerando también el tiempo adicional que se añadiría a la ejecución del algoritmo.

Se propone confeccionar más instancias que varíen en elementos como la cantidad de camiones que se tiene disponibles y los tipos de leche, ya que las instancias tratadas sólo tenían tres tipos de leche y 3 o 100 camiones. También se considera importante generar instancias cuya distancia entre nodos no sea euclidiana, asemejándose más a situaciones de la vida real. Además, se insta a trabajar con más productores de leche para generar más instancias reales.

Finalmente, se propone crear una página web que recopile los mejores resultados para cada instancia, incluyendo elementos como los beneficios, costos y la ruta que realiza cada camión (pudiendo generar una visualización de estas), de tal manera que se puedan corroborar y estudiar todos los elementos de las soluciones y no sólo su calidad final.

REFERENCIAS BIBLIOGRÁFICAS

- [FAO, 2002] FAO (2002). Milk collection, processing and marketing. <http://www.fao.org/3/y3548e/y3548e06.htm>. Último acceso: 15/11/2020.
- [Augerat *et al.*, 1995] Augerat, P., Belenguer, J. M., Benavent, E., Corberán, A., Naddef, D., y Rinaldi, G. (1995). Computational results with a branch and cut code for the capacitated vehicle routing problem. *IMAG*.
- [Basnet *et al.*, 1999] Basnet, C. B., Foulds, L., y Wilson, J. M. (1999). An exact algorithm for a milk tanker scheduling and sequencing problem. *Annals of Operations Research*, 86:559–568.
- [Brown y Graves, 1981] Brown, G. y Graves, G. (1981). Real-time dispatch of petroleum tank trucks. *Management Science*, 27:19–32.
- [Butler *et al.*, 1997] Butler, M., Williams, H. P., y Yarrow, L.-A. (1997). The two-period travelling salesman problem applied to milk collection in Ireland. *Computational Optimization and Applications*, 7(3):291–306.
- [Caramia y Guerriero, 2010] Caramia, M. y Guerriero, F. (2010). A milk collection problem with incompatibility constraints. *Interfaces*, 40:130–143.
- [Chao, 2002] Chao, I.-M. (2002). A tabu search method for the truck and trailer routing problem. *Computers & Operations Research*, 29(1):33 – 51.
- [Chokanat *et al.*, 2019] Chokanat, P., Pitakaso, R., y Sethanan, K. (2019). Methodology to solve a special case of the vehicle routing problem: A case study in the raw milk transportation system. *AgriEngineering*, 1:75–93.
- [Clarke y Wright, 1964] Clarke, G. y Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581.
- [Dantzig y Ramser, 1959] Dantzig, G. B. y Ramser, J. H. (1959). The truck dispatching problem. *Management science*, 6(1):80–91.
- [Dayarian *et al.*, 2015a] Dayarian, I., Crainic, T. G., Gendreau, M., y Rei, W. (2015a). A branch-and-price approach for a multi-period vehicle routing problem. *Computers & Operations Research*, 55:167 – 184.
- [Dayarian *et al.*, 2015b] Dayarian, I., Crainic, T. G., Gendreau, M., y Rei, W. (2015b). A column generation approach for a multi-attribute vehicle routing problem. *European Journal of Operational Research*, 241(3):888 – 906.
- [Dayarian *et al.*, 2016] Dayarian, I., Crainic, T. G., Gendreau, M., y Rei, W. (2016). An adaptive large-neighborhood search heuristic for a multi-period vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review*, 95:95 – 123.

- [Dooley et al., 2005] Dooley, A., Parker, W., y Blair, H. (2005). Modelling of transport costs and logistics for on-farm milk segregation in new zealand dairying. *Computers and Electronics in Agriculture*, 48(2):75 – 91.
- [Fallahi et al., 2008] Fallahi, A. E., Prins, C., y Wolfler Calvo, R. (2008). A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem. *Computers & Operations Research*, 35(5):1725 – 1741. Part Special Issue: Algorithms and Computational Methods in Feasibility and Infeasibility.
- [Fisher, 1994] Fisher, M. L. (1994). Optimal solution of vehicle routing problems using minimum k-trees. *Operations research*, 42(2):626–642.
- [GLH Incorporated, 2014] GLH Incorporated (2014). National milk drug residue data base fiscal year 2014 annual. <http://www.fda.gov/downloads/Food/GuidanceRegulation/GuidanceDocumentsRegulatoryInformation/Milk/UCM434757.pdf>. Último acceso: 15/11/2020.
- [Hoff y Løkketangen, 2008] Hoff, A. y Løkketangen, A. (2008). A tabu search approach for milk collection in western norway using trucks and trailers. In *6th Triennial Symposium on Transportation Analysis, TRISTAN, Pukkett, Thailand*.
- [Igbaria et al., 1996] Igbaria, M., Sprague, R. H., Basnet, C., y Foulds, L. (1996). The impact and benefits of a dss: The case of fleetmanager. *Information & Management*, 31(4):215 – 225.
- [Lahrichi et al., 2015] Lahrichi, N., Crainic, T. G., Gendreau, M., Rei, W., y Rousseau, L.-M. (2015). Strategic analysis of the dairy transportation problem. *Journal of the Operational Research Society*, 66(1):44–56.
- [Lin et al., 2009] Lin, S., Yu, V., y Chou, S. (2009). Solving the truck and trailer routing problem based on a simulated annealing heuristic. *Computers & Operations Research*, 36(5):1683 – 1692.
- [Liu et al., 2008] Liu, S., Lei, L., y Park, S. (2008). On the multi-product packing-delivery problem with a fixed route. *Transportation Research Part E: Logistics and Transportation Review*, 44(3):350–360.
- [Masson et al., 2016] Masson, R., Lahrichi, N., y Rousseau, L.-M. (2016). A two-stage solution method for the annual dairy transportation problem. *European Journal of Operational Research*, 251(1):36 – 43.
- [Paredes-Belmar et al., 2017] Paredes-Belmar, G., Lüer-Villagram, A., Marianov, V., Cortés, C. E., y Bronfman, A. (2017). A milk collection problem with blending. *Computers and Electronics in Agriculture*, 134:109–123–43.
- [Paredes-Belmar et al., 2016] Paredes-Belmar, G., Marianov, V., Bronfman, A., Obreque, C., y Lüer-Villagra, A. (2016). A milk collection problem with blending. *Transportation Research Part E: Logistics and Transportation Review*, 94:26–43.

- [Prasertsri y Kilmer, 2004] Prasertsri, P. y Kilmer, R. L. (2004). Scheduling and routing milk from farm to processors by a cooperative. *Journal of Agribusiness*, 22(2):93S106.
- [Reinelt, 1991] Reinelt, G. (1991). Tsplib—a traveling salesman problem library. *ORSA journal on computing*, 3(4):376–384.
- [Sankaran y Ubgade, 1994] Sankaran, J. K. y Ubgade, R. R. (1994). Routing tankers for dairy milk pickup. *Interfaces*, 24(5):59–66.
- [Sethanan y Pitakaso, 2016] Sethanan, K. y Pitakaso, R. (2016). Differential evolution algorithms for scheduling raw milk transportation. *Computers and Electronics in Agriculture*, 121:245–259.
- [Taillard, 1999] Taillard, E. D. (1999). A heuristic column generation method for the heterogeneous fleet vrp. *Revue française d'automatique, d'informatique et de recherche opérationnelle*, Rech opér 33(1):1–14.
- [Tarantilis y Kiranoudis, 2001] Tarantilis, C. y Kiranoudis, C. (2001). A meta-heuristic algorithm for the efficient distribution of perishable foods. *Journal of Food Engineering*, 50(1):1 – 9.
- [Toth y Vigo, 2002] Toth, P. y Vigo, D. (2002). *The Vehicle Routing Problem*. Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics.
- [Villagrán, 2019] Villagrán, J. (2019). Acercamientos basados en búsqueda local para el problema de recolección de leche con mezcla. *Memoria de titulación*.