

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA  
DEPARTAMENTO DE INFORMÁTICA  
SANTIAGO - CHILE



## “DISEÑO DE UNA TÉCNICA METAHEURÍSTICA PARA EL PROBLEMA DE RECOLECCIÓN DE LECHE CON MEZCLA Y SELECCIÓN”

MATÍAS ALEXANDER MORENO PANDO

MEMORIA PARA OPTAR AL TÍTULO DE  
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Nicolás Rojas Morales  
Profesor Correferente: José Luis Martí

Abril - 2021

## **DEDICATORIA**

Para mis padres María y Hector quienes me dieron la vida, apoyo y educación.

## AGRADECIMIENTOS

En primer lugar agradezco el apoyo incondicional de mis padres María Teresa y Hector, y a mi abuela María Jovita en todo mi proceso de formación universitaria.

Agradezco la paciencia y compañía de mi pareja Geraldine que me enseñó a ver más allá del mundo que conocía.

Agradezco al profesor Nicolás Rojas quien confió en mí cuando más lo necesitaba, me ayudó y motivó en todo el complejo proceso de finalización de una carrera universitaria.

También agradezco al programa de Gratuidad de Educación Superior del ministerio de Educación que me permitió estudiar de manera gratuita en una de las mejores universidades de mi país.

Finalmente, agradezco a todos mis compañeros y amigos de clases, con quienes pasamos tantos buenos y no tan buenos momentos durante toda nuestra carrera universitaria. Se que nuestro futuro será brillante, no por la formación técnica, sino por la calidad humana de cada uno.

## RESUMEN

### Resumen—

Esta memoria aborda el problema de Recolección de Leche mediante técnicas metaheurísticas. Este problema consiste en una planta de producción que requiere un mínimo de recursos para su operación diaria. La planta cuenta con una flota de camiones recolectores para visitar granjas productoras de leche, que se encuentran dispersas en una gran área geográfica. En esta memoria se proponen varias técnicas metaheurísticas que permiten maximizar la diferencia del costo de transporte y el beneficio obtenido en el proceso recolección de leche con selección y mezcla. Los resultados obtenidos son prometedores, al analizar la robustez de la técnica propuesta con instancias de hasta 500 granjas productoras, se obtuvieron nuevos valores máximos con tiempos de ejecución menores a un segundo.

**Palabras Clave—** Enrutamiento de vehículos; Recolección de leche; Metaheurísticas; Optimización.

## ABSTRACT

**Abstract—** This thesis addresses the problem of Milk Collection through metaheuristic techniques. This problem consists of a production plant that requires a minimum of resources for its daily operation. The plant has a fleet of collection trucks to visit dairy farms, which are scattered over a large geographic area. In this thesis, metaheuristic techniques are proposed that allow maximizing the difference in transportation cost and the income obtained in the milk collection process with selection and blending. The results obtained are promising, when analyzing the robustness of the proposed technique with instances of up to 500 producing farms, new maximum values were obtained with execution times less than one second.

**Keywords—** Vehicle routing; Milk collection; Metaheuristics; Optimization.

## **GLOSARIO**

ACS: Ant Colony System

HC: Hill Climbing.

HCFI: Hill Climbing First Improvement.

KP: Knapsack Problem.

MCPSB: Milk Collection Problem with Selection and Blending.

MPVRP: Multiple Product Vehicle Routing Problem.

PCP: Prize Collecting Problem.

PCVRP: Prize Collecting Vehicle Routing Problem.

PDVRP: Pickup and Delivery Vehicle Routing Problem.

SA: Simulated Annealing.

TWVRP: Time Window Vehicle Routing Problem.

TSP: Traveling Salesman Problem.

VRP: Vehicle Routing Problem.

# ÍNDICE DE CONTENIDOS

|                                                                             |           |
|-----------------------------------------------------------------------------|-----------|
| RESUMEN . . . . .                                                           | IV        |
| ABSTRACT . . . . .                                                          | IV        |
| GLOSARIO . . . . .                                                          | V         |
| ÍNDICE DE FIGURAS . . . . .                                                 | VIII      |
| ÍNDICE DE TABLAS . . . . .                                                  | VIII      |
| INTRODUCCIÓN . . . . .                                                      | 1         |
| <b>CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA . . . . .</b>                        | <b>3</b>  |
| 1.1 Contexto y situación Actual . . . . .                                   | 3         |
| 1.2 Modelo matemático . . . . .                                             | 5         |
| 1.2.1 Parámetros . . . . .                                                  | 5         |
| 1.2.2 Variables de decisión . . . . .                                       | 6         |
| 1.2.3 Restricciones . . . . .                                               | 6         |
| 1.3 Objetivos del trabajo . . . . .                                         | 7         |
| 1.3.1 Objetivo General . . . . .                                            | 7         |
| 1.3.2 Objetivos específicos . . . . .                                       | 8         |
| 1.4 Resumen . . . . .                                                       | 8         |
| <b>CAPÍTULO 2: MARCO CONCEPTUAL . . . . .</b>                               | <b>9</b>  |
| 2.1 Técnicas metaheurísticas . . . . .                                      | 9         |
| 2.1.1 Diversificación e intensificación . . . . .                           | 11        |
| 2.1.2 Greedy . . . . .                                                      | 11        |
| 2.1.3 Hill Climbing . . . . .                                               | 13        |
| 2.1.4 Simulated Annealing . . . . .                                         | 14        |
| 2.1.5 Control de temperatura en Simulated Annealing . . . . .               | 16        |
| 2.2 Problemas relacionados . . . . .                                        | 17        |
| 2.2.1 Vehicle Routing Problem - VRP . . . . .                               | 17        |
| 2.3 Prize Collecting Vehicle Routing Problem - PCVRP . . . . .              | 18        |
| 2.3.1 Vehicle Routing Problem with Pick-up and Deliveries - PDVRP . . . . . | 20        |
| 2.3.2 Multi Product Vehicle Routing Problem - MPVRP . . . . .               | 21        |
| 2.3.3 Multi Compartment Vehicle Routing Problem - MCVRP . . . . .           | 23        |
| 2.4 Time Window Vehicle Routing Problem - TWVRP . . . . .                   | 25        |
| 2.4.1 Síntesis . . . . .                                                    | 26        |
| 2.5 Resumen . . . . .                                                       | 26        |
| <b>CAPÍTULO 3: PROPUESTA DE SOLUCIÓN . . . . .</b>                          | <b>28</b> |
| 3.1 Representación de la solución . . . . .                                 | 29        |

|                                                        |                                                                      |           |
|--------------------------------------------------------|----------------------------------------------------------------------|-----------|
| 3.2                                                    | Factibilidad . . . . .                                               | 29        |
| 3.3                                                    | Función de evaluación . . . . .                                      | 30        |
| 3.4                                                    | Inicialización . . . . .                                             | 30        |
| 3.5                                                    | Operadores de Diversificación . . . . .                              | 31        |
| 3.5.1                                                  | Operador Add . . . . .                                               | 32        |
| 3.5.2                                                  | Operador Drop . . . . .                                              | 32        |
| 3.6                                                    | Operadores de Intensificación . . . . .                              | 33        |
| 3.6.1                                                  | Operador Swap Externo . . . . .                                      | 33        |
| 3.6.2                                                  | Operador Swap Interno . . . . .                                      | 34        |
| 3.7                                                    | Metaheurísticas Involucradas . . . . .                               | 34        |
| 3.7.1                                                  | Simulated Annealing . . . . .                                        | 35        |
| 3.7.2                                                  | Hill Climbing First Improvement . . . . .                            | 37        |
| 3.8                                                    | Variantes Propuestas . . . . .                                       | 38        |
| 3.8.1                                                  | Greedy Aleatorizado (G-A) . . . . .                                  | 38        |
| 3.8.2                                                  | Simple Simulated Annealing - SSA . . . . .                           | 39        |
| 3.8.3                                                  | Two Phases Simulated Annealing - 2SA . . . . .                       | 39        |
| 3.8.4                                                  | Improved Two Phases Simulated Annealing - I2SA . . . . .             | 40        |
| 3.8.5                                                  | Two Phases Routing Algorithm - 2PRA . . . . .                        | 41        |
| 3.9                                                    | Cambio de etapas . . . . .                                           | 42        |
| 3.9.1                                                  | Cantidad fija de iteraciones . . . . .                               | 42        |
| 3.9.2                                                  | Cantidad de iteraciones sin mejora . . . . .                         | 43        |
| 3.9.3                                                  | Cantidad de iteraciones desequilibradas . . . . .                    | 43        |
| 3.10                                                   | Conclusiones . . . . .                                               | 43        |
| <b>CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN . . . . .</b> |                                                                      | <b>45</b> |
| 4.1                                                    | Instancias . . . . .                                                 | 45        |
| 4.1.1                                                  | Instancias reales . . . . .                                          | 45        |
| 4.1.2                                                  | Instancias de prueba . . . . .                                       | 46        |
| 4.2                                                    | Entorno Experimental . . . . .                                       | 47        |
| 4.3                                                    | Parámetros . . . . .                                                 | 47        |
| 4.4                                                    | Sintonización . . . . .                                              | 48        |
| 4.5                                                    | Resultados de las versiones . . . . .                                | 49        |
| 4.6                                                    | Comparación con Técnicas Existentes . . . . .                        | 51        |
| 4.7                                                    | Convergencia y análisis de scheduling de Temperatura . . . . .       | 55        |
| 4.8                                                    | Efecto de las distintas etapas en las rutas de recolección . . . . . | 60        |
| 4.9                                                    | Conclusiones . . . . .                                               | 63        |
| <b>CAPÍTULO 5: CONCLUSIONES . . . . .</b>              |                                                                      | <b>65</b> |
| <b>REFERENCIAS BIBLIOGRÁFICAS . . . . .</b>            |                                                                      | <b>67</b> |

# ÍNDICE DE FIGURAS

|    |                                                                                                                                 |    |
|----|---------------------------------------------------------------------------------------------------------------------------------|----|
| 1  | Ejemplo de escenario para Milk Collection Problem. . . . .                                                                      | 3  |
| 2  | Posibles rutas para escenarios con y sin mezcla permitida. . . . .                                                              | 4  |
| 3  | Ejemplo de solución para MCP. Fuente: Elaboración propia . . . . .                                                              | 11 |
| 4  | Ejemplo de Knapsack Problem. Fuente: [Wikipedia, ] . . . . .                                                                    | 12 |
| 5  | Comportamiento de $P$ en función de la temperatura. . . . .                                                                     | 16 |
| 6  | Comparación visual de TSP (izquierda) y VRP (derecha). . . . .                                                                  | 18 |
| 7  | Distribución de granjas (a) y puntos de acopio (b). . . . .                                                                     | 21 |
| 8  | Comparación de rutas sin mezclar calidades (b) y con posibilidad de mezcla (c). . . . .                                         | 22 |
| 9  | Comparación visual de solución sin clusters (izquierda) y con clusters (derecha). Fuente: [Reed <i>et al.</i> , 2014] . . . . . | 25 |
| 10 | Operador Add antes (a) y después (b) del movimiento aplicado. . . . .                                                           | 32 |
| 11 | Operador Swap Externo antes (a) y después (b) del movimiento aplicado. . . . .                                                  | 34 |
| 12 | Operador Swap Interno antes (a) y después (b) del movimiento aplicado. . . . .                                                  | 34 |
| 13 | Ejemplo de scheduling exponencial (a), lineal (b) y sinusoidal (c). . . . .                                                     | 37 |
| 14 | Estructura general de técnica 2PRA. Fuente: Elaboración propia. . . . .                                                         | 41 |
| 15 | Resultados de instancia r500 en diferentes tiempos de ejecución. . . . .                                                        | 56 |
| 16 | Valor de solución actual durante dos ciclos en la ejecución de SSA. . . . .                                                     | 57 |
| 17 | Valor de solución actual durante dos ciclos en la ejecución de 2SA. . . . .                                                     | 58 |
| 18 | Valor de solución actual durante dos ciclos en la ejecución de I2SA. . . . .                                                    | 59 |
| 19 | Valor de solución actual durante dos ciclos en la ejecución de 2PRA. . . . .                                                    | 60 |
| 20 | Rutas generadas a partir del algoritmo G-A. . . . .                                                                             | 61 |
| 21 | Configuración de rutas luego de la etapa de diversificación. . . . .                                                            | 62 |
| 22 | Configuración de rutas luego de la etapa de intensificación. . . . .                                                            | 63 |

## ÍNDICE DE TABLAS

|    |                                                                                                                |    |
|----|----------------------------------------------------------------------------------------------------------------|----|
| 1  | Tabla resumen variantes VRP . . . . .                                                                          | 27 |
| 2  | Detalle de instancias reales . . . . .                                                                         | 46 |
| 3  | Detalle de instancias de prueba . . . . .                                                                      | 47 |
| 4  | Parámetros utilizados para la comparación de las distintas versiones . . . . .                                 | 48 |
| 5  | Resultados de las técnicas implementadas . . . . .                                                             | 49 |
| 6  | Parámetros utilizados en los experimentos finales . . . . .                                                    | 51 |
| 7  | Mejores resultados CPLEX vs MCSA vs 2PRA en instancias de prueba, Gap de mejores resultados . . . . .          | 52 |
| 8  | Tiempos de ejecución CPLEX vs MCSA vs 2PRA en instancias de prueba . . . . .                                   | 53 |
| 9  | Mejores resultados MCSA vs 2PRA en instancias reales, Gap de resultados promedio . . . . .                     | 53 |
| 10 | Tiempos de ejecución MCSA vs 2PRA en instancias reales . . . . .                                               | 54 |
| 11 | Mejores resultados MCSA vs 2PRA en instancias reales sin mínimo de leche, Gap de resultados promedio . . . . . | 54 |
| 12 | Tiempos de ejecución MCSA vs 2PRA en instancias reales sin mínimo de leche . . . . .                           | 55 |

## INTRODUCCIÓN

Desde hace un tiempo, los modelos matemáticos son usados para optimizar procesos industriales [Yu y Yang, 1998] y de logística [Sbihi y Eglese, 2007]. En el caso de logística, se presentan diversos problemas de gran complejidad. Por ejemplo, se tiene una empresa encargada de distribuir y recolectar productos con un conjunto de vehículos. Si se busca maximizar los beneficios de la empresa, las distancias recorridas por los vehículos deben ser minimizadas [Toth y Vigo, 2014]. En este contexto aparecen una variedad de problemas conocidos como Vehicle Routing Problem (VRP). Este problema responde a encontrar el mejor conjunto de rutas que permiten satisfacer una necesidad, como por ejemplo, recoger y entregar encomiendas, recolección y transporte de materiales peligrosos, entrega de materias primas a empresas productoras, entre otros. En el capítulo 2 se presenta una revisión de algunas variantes del VRP.

El transporte de materia prima es una de las primeras actividades en las cadenas de producción. Esto generalmente involucra vehículos con gran capacidad, que producen altos costos de combustible y mantenimiento. Para el caso de la industria de lácteos estos costos son significativos. Más de un 30 % del costo total de la producción leche corresponde a la recolección y transporte [FAO, 2016]. En Chile, existen empresas que compran y recolectan leche de productores locales, la cual es almacenada, procesada y utilizada en plantas de fabricación de productos lácteos. La mayor parte de la leche producida a nivel nacional se concentra en la zona sur del país. Parte de la producción de leche proviene de pequeñas granjas que se localizan dispersas en una gran área geográfica que venden sus producciones individuales o en forma de cooperativas a empresas de productos lácteos.

Las plantas recolectoras de leche disponen de flotas de camiones de transporte que recorren cientos de kilómetros diarios. En su ruta, los camiones recolectan leche desde granjas de producción, para finalmente volver a la planta. En casos reales del sur de Chile, existen plantas con decenas de camiones recolectores que deben visitar cientos de granjas para conseguir la cantidad mínima que necesita para la fabricación de productos lácteos. Esto genera la necesidad de encontrar rutas de recolección eficientes, que permitan obtener la cantidad suficiente de leche y que las distancias recorridas por los camiones sean mínimas, buscando disminuir los costos de transporte. Esto da origen al Milk Collection Problem (MCP), problema de optimización combinatoria que consiste en maximizar los beneficios obtenidos en el proceso de recolección de leche. Este trabajo aborda una variante del MCP llamada Milk Collection Problem with Selection and Blending (MCPSB) que será presentada en el capítulo 2.

Para encontrar soluciones a este problema de optimización combinatoria se propone el diseño e implementación de varias técnicas metaheurísticas, que permiten resolver el problema en una cantidad acotada de tiempo. Las técnicas metaheurísticas utilizadas en este trabajo serán descritas al final del capítulo 2. Mientras que, en el capítulo 3 se proponen técnicas heurísticas para resolver el MCPSB.

En el capítulo 4 se realizan experimentos con las distintas técnicas metaheurísticas diseñadas, se implementan nuevos algoritmos de búsqueda, se analizan los resultados, y se implementan iterativamente mejoras a las técnicas para enfrentar los problemas y desafíos encontrados. Finalmente en el capítulo 5 se concluye describiendo los alcances y limitaciones de la técnica propuesta, y se identifican las principales contribuciones y aplicaciones del trabajo realizado.

## CAPÍTULO 1

### DEFINICIÓN DEL PROBLEMA

#### 1.1. Contexto y situación Actual

En el sur de Chile existe una gran industria de producción lechera. Aquí, las plantas procesadoras de lácteos compran y recolectan leche para satisfacer la cantidad de leche que necesitan para la fabricación de productos. Las plantas procesadoras tienen una flota de camiones recolectores que transportan la leche desde las granjas productoras hasta la planta. Estas granjas se encuentran dispersas en un gran área geográfica, por lo que se necesitan rutas de recolección para cada camión de tal forma que puedan recolectar la leche de manera eficiente, minimizando las distancias recorridas y maximizando el beneficio obtenido. El escenario geográfico puede ser representado de manera visual como un grafo de nodos y arcos, donde los nodos representan a la planta procesadora y a las granjas junto con su producción, mientras que los arcos representan los caminos y distancias entre cada nodo. Esto se ilustra en la Figura 1.

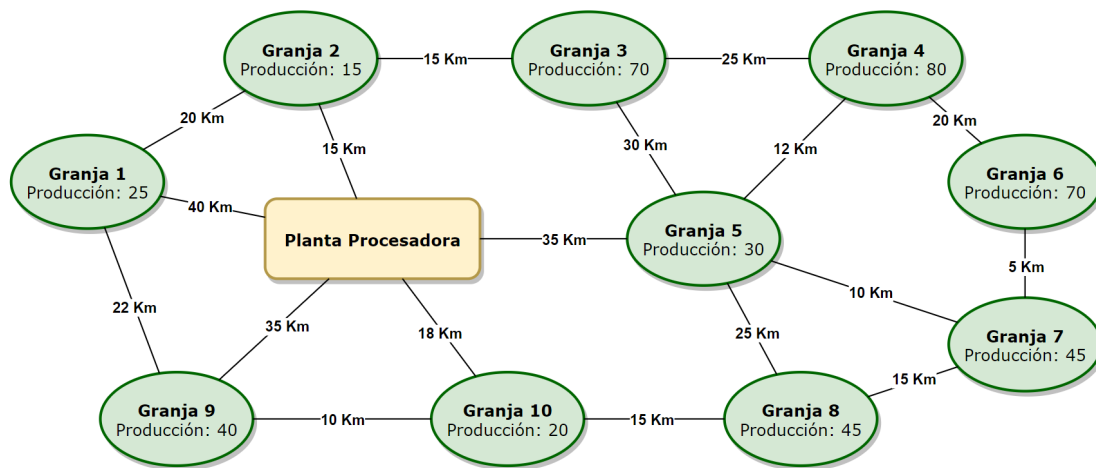


Figura 1: Ejemplo de escenario para Milk Collection Problem.

Fuente: Elaboración propia.

Se hace la diferencia entre dos modelos de producción y recolección. En el primero, la planta procesadora recolecta diariamente toda la leche de las granjas productoras, lo que da origen al Milk Collection Problem (MCP) [Rojas Molina, 2005]. Aquí, se buscan las rutas de recolección con el fin de minimizar el costo de transporte asociado. En el segundo modelo, la planta procesadora selecciona y visita las granjas productoras necesarias para recolectar su producción de leche. En este segundo caso, se busca maximizar el beneficio obtenido por la leche obtenida y minimizar el costo de transporte asociado, lo que da origen al Milk Collection Problem with Selection (MCPS) [Montero *et al.*, 2019].

La leche producida en distintas granjas tiene distintas calidades, las cuales se pueden clasificar para utilizar en la fabricación de distintos productos. Si se tienen camiones de solo un compartimento, cada camión recolector puede almacenar solo una calidad de leche en una ruta de recolección, o puede mezclar distintas calidades de leche en un solo camión. En la Figura 2 se muestra un escenario de una planta y 3 camiones recolectores. En el lado izquierdo de la figura se muestran las rutas (prohibiendo la mezcla de distintas calidades), y en contraste, al lado derecho se muestran posibles rutas si la mezcla de distintas calidades es permitida. La cantidad de rutas posibles aumenta cuando es posible mezclar leche de distintas calidades.

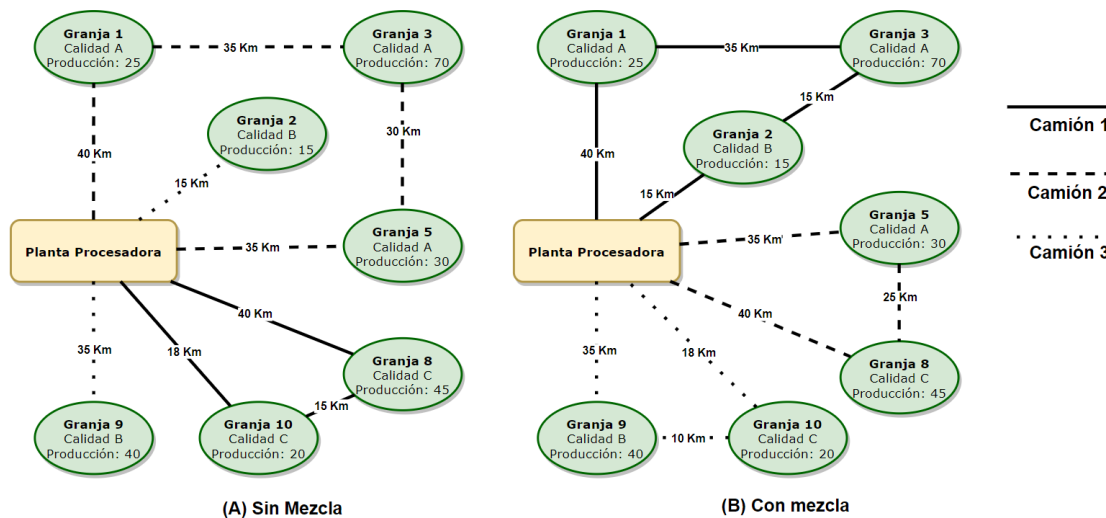


Figura 2: Posibles rutas para escenarios con y sin mezcla permitida.

Fuente: Elaboración propia.

La mezcla de leche permite encontrar rutas que otorguen un mayor beneficio, lo que da origen al Milk Collection Problem with Selection and Blending (MC-PSB) [Paredes-Belmar *et al.*, 2017]. El MCPSB es un problema de optimización combinatoria que busca obtener rutas de recolección para cada camión, permitiendo la mezcla de distintas calidades de leche, además de maximizar el beneficio por la leche recolectada y minimizar el costo de transporte asociado.

Para la versión del MCPSB que se aborda en esta memoria se tiene un conjunto de camiones, un conjunto de granjas productoras de leche y una cantidad mínima de cada calidad de leche que requiere la planta procesadora. Aquí, es posible recolectar más leche de la mínima necesaria con el fin de generar un beneficio mayor. Los camiones tienen solo un compartimento y poseen la misma capacidad de carga. Además, el costo de transporte tiene una relación directamente proporcional con la distancia recorrida por los camiones. Para cada camión se busca una ruta óptima, en la que se presenta el orden de las granjas que debe visitar. Cuando el camión visita una granja, éste debe recolectar toda su producción de leche. Todos los camiones inician su ruta en la planta y deben regresar al visitar todas las granjas. En cuanto a la demanda de leche, se tienen tres calidades distintas de leche que tienen un beneficio

asociado, primero la calidad A (la más alta), luego la calidad B y finalmente la calidad C (la más baja). Cuando se mezclan dos o más calidades de leche, la calidad de la mezcla resultante equivale a la menor calidad presente en la mezcla. Además, cada granja produce solo una calidad de leche y puede ser visitada solo una vez. Por último, Los camiones pueden mezclar distintas calidades de leche en su compartimento y también la leche puede ser mezclada una vez que llega a la planta, lo que se denomina mezcla en planta.

## 1.2. Modelo matemático

A continuación se presenta un modelo matemático que describe el MCPSB, basado en el trabajo de Paredes [Paredes-Belmar *et al.*, 2016].

### 1.2.1. Parámetros

Se definen los siguientes componentes que serán utilizados en posteriores ecuaciones:

- $F$ : Conjunto de nodos (granjas y planta), donde el nodo 0 corresponde a la planta
- $Q$ : Conjunto de calidades de leche
- $P_q^f$ : Cantidad de leche de calidad  $q$  producida por granja  $f$ ,  $\forall f \in F, \forall q \in Q$
- $T$ : Conjunto de camiones
- $L_t$ : Carga máxima del camión  $t$ ,  $\forall t \in T$
- $V_q$ : Beneficio por litro de leche de calidad  $q$
- $G_q$ : Conjunto de granjas que producen leche de calidad  $q$ ,  $\forall q \in Q$
- $C_{ij}$ : Costo de transporte entre las granjas  $i$  y  $j$ ,  $\forall i, j \in F$
- $D_q$ : Demanda de la planta de calidad de leche  $q$ ,  $\forall q \in Q$

### 1.2.2. Variables de decisión

Se definen las siguientes variables de decisión para el problema.

$$\begin{aligned}
 x_{ij}^t &= \begin{cases} 1 & \text{si el camión } t \text{ viaja del nodo } i \text{ al } j \\ 0 & \text{si no lo hace} \end{cases} \\
 y_i^{tq} &= \begin{cases} 1 & \text{si el camión } t \text{ carga leche de calidad } q \\ & \text{desde la granja } i \\ 0 & \text{si no lo hace} \end{cases} \\
 z^{tq} &= \begin{cases} 1 & \text{si camión } t \text{ entrega leche de calidad } q \\ 0 & \text{si no lo hace} \end{cases} \\
 w^{tq} &= \text{Volumen de calidad } q \text{ que el camión } t \text{ entrega} \\
 &\quad \text{a la planta} \\
 v^{qr} &= \text{Volumen de calidad } q \text{ entregado a la planta} \\
 &\quad \text{mezclado para uso de calidad } r
 \end{aligned}$$

El objetivo de este problema es encontrar rutas que permitan maximizar los beneficios percibidos por la recolección de leche. La función objetivo es:

$$fo = \text{Max} \sum_{t \in T} \sum_{q \in Q} V_q z_{tq} - \sum_{t \in T} \sum_{(i,j) \in F} x_{ij}^t C_{ij}$$

La función objetivo comprende dos grandes componentes, minimizar el costo asociado al transporte de leche, y por otro lado, maximizar las ganancias obtenidas por la leche recolectada y mezclada.

### 1.2.3. Restricciones

Un conjunto de rutas correcto deben cumplir las siguientes restricciones:

1. La cantidad de leche transportada por un camión no debe superar su capacidad máxima.

$$\sum_{q \in Q} \sum_{f \in F} \sum_{p \in P} p_q^f y_f^{tq} \leq L_t \quad \forall t \in T$$

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

$$\sum_{t \in T} \sum_{q \in Q} y_f^{tq} \leq 1 \quad \forall f \in F$$

3. Todos los camiones inician y finalizan su ruta en la planta.

$$\sum_{0,j \in F} x_{0j}^t = 1 \quad \forall t \in T$$

$$\sum_{i,0 \in F} x_{i0}^t = 1 \quad \forall t \in T$$

4. No hay saltos en el recorrido de los camiones.

$$\sum_{i,j \in F} x_{ij}^t = \sum_{j,h \in F} x_{jh}^t \quad \forall t \in T$$

5. Se deben cumplir las demandas mínimas de leche de la planta.

$$\sum_{q \in Q} v^{qr} \geq D_r \quad \forall r \in D_r$$

6. Toda leche recolectada se entrega a la planta.

$$\sum_{q \in Q} \sum_{f \in F} \sum_{p \in P} p_q^f y_f^{tq} = \sum_{q \in Q} \sum_{t \in T} w^{tq} \quad \forall t \in T$$

7. La leche que se entrega en la planta puede ser mezclada.

$$\sum_{r \in D} v^{qr} \geq \sum_{t \in T} w^{tq} \quad \forall q \in Q$$

8. La leche entregada por un camión es considerada de una sola calidad.

$$\sum_{q \in Q} z^{tq} \leq 1 \quad \forall t \in T$$

## 1.3. Objetivos del trabajo

### 1.3.1. Objetivo General

Diseñar una técnica metaheurística para resolver el problema de recolección de leche con selección y mezcla de manera eficiente, en términos de la calidad de las soluciones o de los recursos utilizados.

### **1.3.2. Objetivos específicos**

- Realizar una revisión de técnicas existentes en la literatura diseñadas para resolver el problema de recolección de leche
- Diseñar y proponer técnicas para resolver el problema de recolección de leche con selección y mezcla
- Analizar el efecto de cada componente de las técnicas propuestas en términos de la calidad de las soluciones encontradas.
- Comparar el desempeño de las técnicas propuestas con el estado del arte, utilizando instancias benchmark y análisis estadístico.

### **1.4. Resumen**

En este capítulo se define el contexto y el problema a abordar en esta memoria. El problema de recolección de leche con selección y mezcla consiste en maximizar el beneficio obtenido por una planta procesadora de productos lácteos. La planta recolecta la leche con una flota de camiones homogéneos de un solo compartimento. Para cada camión se diseña una ruta que describe las granjas que debe visitar para recolectar la leche. El problema considera tres calidades de leche, cada una con un valor asociado. Se permite la mezcla de leche con el costo de pérdida de calidad, pero con el beneficio de obtener rutas más eficientes de recolección. El objetivo del problema es encontrar rutas óptimas para cada camión con el fin de maximizar el beneficio obtenido por la recolección de leche y minimizar el costo de transporte asociado.

## CAPÍTULO 2

### MARCO CONCEPTUAL

Milk Collection Problem with Selection and Blending (MCPSB) es un problema de optimización combinatoria que tiene por objetivo satisfacer las restricciones de transporte y demanda de distintas calidades de leche para una planta procesadora, buscando minimizar el costo requerido para la recolección y maximizar el beneficio obtenido por la recolección de leche.

Si consideramos una instancia donde un camión debe visitar 15 estaciones, el espacio de búsqueda que considera todas las posibles rutas tiene por tamaño  $6,53 \cdot 10^{11}$  soluciones candidatas. Así, un algoritmo de búsqueda completa (por ejemplo, Generate and Test) debería evaluar todas las posibles combinaciones. Esto provoca que cuando aumenta la cantidad de estaciones que se pueden visitar, el número de combinaciones crece exponencialmente, lo que resulta en una tarea imposible de realizar en un tiempo de computación acotado.

Una forma de encontrar soluciones a problemas complejos es utilizar técnicas de búsqueda incompleta/local [Aarts *et al.*, 2003]. Las técnicas de búsqueda incompleta son métodos que permiten resolver problemas de optimización combinatoria, optimización continua y problemas de múltiples objetivos, en un tiempo de computación acotado. En muchos casos, se sacrifica obtener una solución globalmente óptima, por obtener soluciones de calidad aceptable en menos tiempo. El objetivo es encontrar el mejor conjunto de valores, para todas las variables del problema, que permitan satisfacer las restricciones del problema en particular y que a su vez, optimice el (los) objetivo(s) en cuestión.

En este capítulo se describe la base conceptual, las herramientas y metodologías necesarias para abordar el MCPSB. También se presenta una revisión de trabajos presentes en la literatura que tienen relación con el MCPSB, con el objetivo de estudiar distintas técnicas que han sido utilizadas para encontrar soluciones a problemas de optimización de rutas y recolección de bienes.

#### 2.1. Técnicas metaheurísticas

En el contexto de optimización combinatoria, se definen los siguientes conceptos:

- *Función Objetivo*: Función a optimizar en el problema de optimización.
- *Función de Evaluación*: Función utilizada para guiar el proceso de búsqueda de las técnicas de resolución. En general, mide la calidad de las soluciones y podría ser igualmente definida a la Función Objetivo.
- *Solución Factible*: Un conjunto de valores para las variables del problema que satisfacen las restricciones de un problema de optimización.

- *Solución Infactible*: Un conjunto de valores para las variables del problema, pero que no satisfacen una o más restricciones del problema de optimización.
- *Espacio de soluciones*: El conjunto de todas las soluciones posibles para un problema de optimización, tanto factibles como infactibles.
- *Movimiento*: Modificación o perturbación que se aplica a una solución.
- *Vecindario*: Conjunto de soluciones que pueden ser alcanzadas tras la aplicación secuencial de un movimiento a una solución en particular.
- *Óptimo Local*: Una solución es un Óptimo local si su valor de la función objetivo es la menor en comparación a las soluciones de su vecindario.<sup>1</sup>
- *Óptimo Global*: Una solución es un Óptimo global si su valor en la función objetivo es la menor con respecto a todo el espacio de soluciones.
- *Técnica de Búsqueda Completa*: Algoritmo que realiza una búsqueda exhaustiva en el espacio de soluciones y permite encontrar una (o todas las) solución(es) óptima(s) del problema o verificar que el problema no tiene solución.
- *Técnica de Búsqueda Incompleta*: Algoritmo que realiza una búsqueda por regiones del espacio de soluciones, con el objetivo de encontrar soluciones cercanas al óptimo en un tiempo de computación acotado.

Por ejemplo, una solución para el Milk Collection Problem (MCP) es un conjunto de rutas para cada camión que permitan satisfacer las restricciones del problema. Mientras que, la calidad de la solución sería el valor de la función de evaluación que representa el beneficio obtenido por la empresa por la leche recolectada menos el costo de transporte. En la figura 3 se observa una representación de una solución. En este caso, se tienen 2 camiones de 250 litros de capacidad, y se requieren 240 litros de leche de calidad A y B. El costo de transporte se representa en las flechas que unen los nodos y el beneficio obtenido por la leche de calidad A es de 2 unidades por litro, y en el caso de la leche de calidad B, el beneficio es de 1 unidad por litro. Las granjas de producción de leche de calidad A son representadas por cuadrados de borde continuo, y las granjas de calidad B por cuadrados de borde segmentado. Una solución posible para este escenario es la siguiente. El camión 1 (flecha continua), parte desde la planta hacia el nodo 1, luego viaja al nodo 2 y vuelve a la planta, el costo de transporte es de 75, y se recolectan 245 litros de leche de calidad A, por un beneficio de 490. Por otro lado, el camión 2, inicia su recorrido desde la planta hacia el nodo 7, luego viaja al nodo 4 y termina su recorrido en la planta, el costo de transporte es de 60, se recolectan 240 litros de leche de calidad B por un beneficio de 240. La utilidad consiste en el beneficio obtenido por la leche recolectada, menos el costo de transporte, lo que resulta una utilidad de 595 que representa el valor de la función objetivo del problema. Para este caso, esta solución al problema representa el óptimo global.

---

<sup>1</sup>En un problema de maximización se busca la solución con el mayor valor de la función objetivo

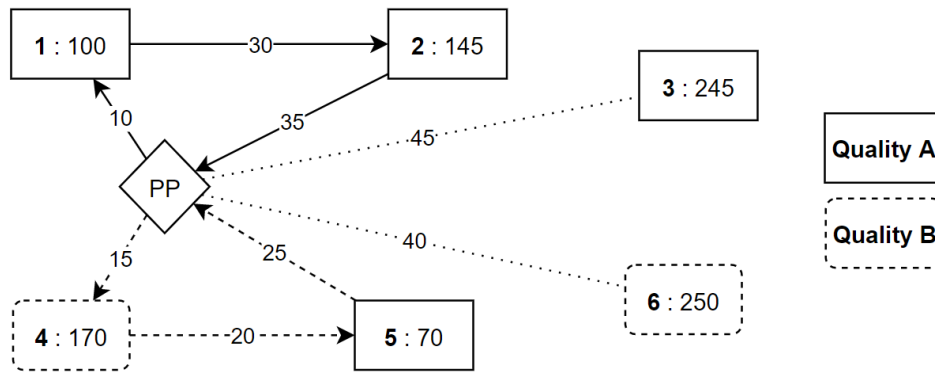


Figura 3: Ejemplo de solución para MCP. Fuente: Elaboración propia

Uno de los métodos que utilizan búsqueda incompleta para encontrar soluciones a los problemas de optimización mencionados son las técnicas metaheurísticas. Estas se definen como métodos de solución que orquestan una interacción entre procedimientos de búsqueda incompleta y estrategias de alto nivel, para crear un proceso capaz de escapar de óptimos locales y ejecutar una búsqueda robusta en un espacio de solución [Glover y Kochenberger, 2006].

### 2.1.1. Diversificación e intensificación

Las técnicas heurísticas permiten una búsqueda eficaz y efectiva en un gran espacio de soluciones para un problema particular. Esta búsqueda puede ser separada en dos etapas: intensificación (o explotación), y por otro lado, diversificación (o exploración). La etapa de intensificación realiza una búsqueda focalizada en una región del espacio de soluciones. Comúnmente el objetivo de esta etapa es encontrar la mejor solución de una región. Mientras que la etapa de diversificación realiza una búsqueda más amplia en el espacio de soluciones que permite escapar de máximos locales y encontrar otras regiones del espacio de soluciones para realizar nuevamente una etapa de intensificación. Estas etapas suelen intercalarse una después de la otra, cuando se agota la búsqueda por intensificación, es decir, la calidad de las soluciones no mejoran según un criterio de detención, la etapa de diversificación se encarga de llevar la búsqueda a otra región. Un buen equilibrio entre esos dos tipos de búsqueda es clave en el éxito de una técnica metaheurística. A continuación, se presentarán detalles de algunas técnicas metaheurísticas existentes en la literatura.

### 2.1.2. Greedy

Greedy es una técnica que permite construir soluciones de manera rápida para problemas de optimización. Esta técnica consiste en construir soluciones añadiendo, en cada paso, el

mejor elemento posible según una función miope. Los componentes de la técnica Greedy se detallan a continuación.

- *Representación*: Estructura que representa una solución del problema.
- *Función Miope*: Función que selecciona el siguiente elemento para añadir a la solución.
- *Punto de Partida*: Solución inicial desde donde comienza la construcción.
- *Función de Evaluación*: Función que determina numéricamente la calidad de la solución.

Esta selección del “Mejor elemento” puede conducir a la creación de soluciones de mala calidad, dado que el algoritmo puede elegir elementos que parecen muy buenos en el momento, pero en la perspectiva de la solución completa puede resultar en malas elecciones. Esta técnica puede ser utilizada en el Knapsack Problem (KP), el cual es un problema de optimización combinatoria. Aquí, se tiene una serie de objetos de entre los cuales se debe seleccionar un subconjunto de ellos. Cada objeto genera una ganancia asociada y tiene un peso determinado. El subconjunto de objetos seleccionados no debe superar la capacidad de la mochila y el objetivo es maximizar la ganancia total. La figura 4 ilustra un ejemplo de instancia del KP.

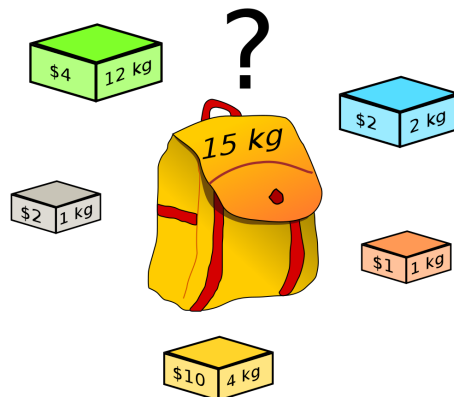


Figura 4: Ejemplo de Knapsack Problem. Fuente: [Wikipedia, ]

Para utilizar Greedy para resolver KP, se pueden definir los componentes de la siguiente manera:

- *Representación*: Lista de bits, donde cada bit representa si el objeto  $i$  se lleva en la mochila,  $[x_0, x_1, \dots, x_n]$  con  $n$  igual al número de objetos totales.
- *Función Miope*: Añadir el objeto de mayor valor a la mochila sin exceder el peso máximo (factible).

- *Punto de Partida:* Lista de bits vacía, es decir, la mochila vacía  $[0, 0, \dots, 0]$ .
- *Función de Evaluación:* Suma de los valores de los objetos presentes en la mochila.

En cada iteración se escoge el objeto de mayor valor según la función miope hasta que se alcanza el límite de peso. Con los valores del ejemplo de la figura 7, el primer objeto seleccionado es el objeto amarillo, luego el objeto gris (o azul), luego el objeto azul (o gris) y finalmente el objeto naranja en donde se llega al límite de peso. En este ejemplo la solución encontrada coincide con el óptimo global.

### 2.1.3. Hill Climbing

Una técnica de búsqueda incompleta que se utiliza en esta memoria es llamada Hill Climbing (HC). HC es una técnica de optimización que permite realizar búsqueda incompleta en un espacio de soluciones. HC es un algoritmo iterativo que intenta encontrar soluciones realizando movimientos en la solución actual, incrementando iterativamente la calidad de la solución. Esta técnica solo permite mejoras en la calidad de la solución actual, lo que puede ser imaginado como la subida de un cerro hasta una de las cimas (máximos locales). En HC, en cada iteración de la técnica se analizan los vecinos de la solución actual con el objetivo de encontrar un vecino con un mejor valor de la función de evaluación. Si no se encuentra un vecino con una mejor función objetivo, el algoritmo se detiene y se clasifica la solución actual como un máximo local.

Dos variantes de HC son Hill Climbing First Improvement (HCFI) y Hill climbing Best Improvement (HCBI). En el algoritmo 1 se detalla una implementación de HCFI, mientras que en el algoritmo 2 se detalla una implementación de HCBI.

---

**Algorithm 1:** Hill Climbing First Improvement

---

**Result:**  $S_{current}$   
 $S_{current} \leftarrow \text{randomInitialSolution}();$   
**for**  $i \leftarrow 0$  **to**  $SizeNeigh$  **do**  
     $S_{candidate} \leftarrow \text{randomNeighbor}(S_{current});$   
    **if**  $eval(S_{candidate}) > eval(S_{current})$  **then**  
         $S_{current} \leftarrow S_{candidate};$   
    **end**  
**end**  
**return**  $S_{current};$

---

---

**Algorithm 2:** Hill Climbing Best Improvement

---

**Result:**  $S_{current}$   
 $S_{current} \leftarrow \text{randomInitialSolution}();$   
**for**  $i \leftarrow 0$  **to**  $SizeNeigh$  **do**  
     $S_{candidate} \leftarrow \text{bestNeighbor}(S_{current});$   
    **if**  $eval(S_{candidate}) > eval(S_{current})$  **then**  
         $S_{current} \leftarrow S_{candidate};$   
    **else**  
        **return**  $S_{current}$   
    **end**  
**end**  
**return**  $S_{current};$

---

La principal diferencia que se puede notar en los algoritmos, es que en HCFI es necesario encontrar el primer vecino que tenga una mejor función de evaluación para realizar el movimiento. Mientras que en HCBI se revisa completamente todo el vecindario de soluciones para seleccionar el vecino con mejor función de evaluación. Se alcanza un máximo local cuando no se encuentra un vecino con una mejor función de evaluación que la solución actual.

#### 2.1.4. Simulated Annealing

Una técnica metaheurística que se aborda en esta memoria es llamada Simulated Annealing (SA) [Van Laarhoven y Aarts, 1987] la cual se inspira en el proceso del recocido de metales. Una de los usos de esta técnica sirve para encontrar soluciones en problemas de optimización combinatoria. La idea fundamental dentro de SA es permitir movimientos que resulten en soluciones de peor calidad que la solución actual con el objetivo de escapar de máximos locales. La probabilidad de realizar un movimiento que empeora la calidad disminuye durante

la búsqueda. En el algoritmo 3 se representa una implementación de SA.

---

**Algorithm 3:** Simulated Annealing

---

**Result:** Best Solution Found  
 $S_{curr} \leftarrow \text{randomInitialSolution}();$   
 $S_{best} \leftarrow S_{curr};$   
**for**  $i \leftarrow 0$  **to**  $iteration_{max}$  **do**  
     $S_{new} \leftarrow \text{createNeighborSolution}(S_{curr});$   
     $temp \leftarrow \text{calculateTemperature}(i);$   
    **if**  $eval(S_{new}) \geq eval(S_{curr})$  **then**  
         $S_{curr} \leftarrow S_{new};$   
        **if**  $eval(S_{new}) \geq eval(S_{best})$  **then**  
             $S_{best} \leftarrow S_{new};$   
        **end**  
    **end**  
    **else if**  $\exp(\frac{eval(S_{curr}) - eval(S_{new})}{temp}) > rand(0,1)$  **then**  
         $S_{curr} \leftarrow S_{new};$   
    **end**  
**end**  
**return**  $S_{best};$

---

SA comienza con una solución inicial construida de manera aleatoria o heurísticamente, y se inicializa el parámetro de temperatura  $temp$ . En cada iteración de SA, se aplican movimientos a la solución actual  $S_{curr}$  del problema generando una nueva solución  $S_{new}$ . Estos movimientos siempre son aceptados si la nueva solución  $S_{new}$  tiene un mejor valor en la función de evaluación. En caso contrario, si el movimiento genera un empeoramiento en la calidad, se utiliza una función que retorna una probabilidad  $P$  de aceptación del movimiento el cual dependerá de la temperatura del sistema  $temp$  y de la magnitud del empeoramiento  $\Delta_{eval}$ . La probabilidad generalmente se calcula generalmente utilizando la distribución de de probabilidad Boltzmann. En la ecuación 1 se representa la función de probabilidad de aceptación para el caso de un problema de maximización.

$$P = e^{\frac{\Delta_{eval}}{T}} \quad \Delta_{eval} = f(S_{new}) - f(S_{curr}) \quad (1)$$

De esta forma, para el caso de una nueva solución ( $s^*$ ) que tenga una menor calidad, la probabilidad de aceptación es mayor cuando la temperatura es alta, y si la temperatura es baja la probabilidad de aceptación disminuye. En SA se realizan miles o millones de iteraciones durante la ejecución de la técnica mientras que la temperatura varía durante el proceso, con el objetivo de encontrar soluciones al problema. Al inicio de la ejecución la temperatura generalmente es alta y permite una exploración de varias zonas en el espacio de búsqueda. Luego, la temperatura disminuye para dar paso a una etapa de explotación en donde prin-

principalmente se aceptan movimientos que buscan mejorar la calidad de la solución en cada iteración.

### 2.1.5. Control de temperatura en Simulated Annealing

En Simulated Annealing se necesita un mecanismo de control de temperatura, con el objetivo de regular la probabilidad de aceptación de un movimiento que empeore la calidad de la solución actual. Este mecanismo de control de temperatura es clave para explorar de manera eficiente diversas regiones del espacio de soluciones y para explotar una región en búsqueda de soluciones de mayor calidad. Cuando la temperatura es alta, es más probable que un movimiento que empeore la solución actual sea aceptado. Mientras que, cuando la temperatura es baja, es poco probable que se acepte un movimiento que disminuya la calidad de la solución. En la figura 5 en el eje vertical, se muestra el valor de  $P$ , que corresponde a la función de probabilidad de aceptación, y en el eje horizontal se indica la temperatura del sistema. En este caso, el valor de  $S_{curr}$  es 3500, y  $S_{new}$  es 3490, es decir, la nueva solución tiene peor calidad que la solución actual. Se observa que existe una relación directa entre la temperatura y la probabilidad de aceptación  $P$ . Si la temperatura tiene un valor de 5, la probabilidad de aceptar el movimiento es de solo un 13 %. Por otro lado, si la temperatura tiene un valor de 30, la probabilidad aumenta a un 72 %.

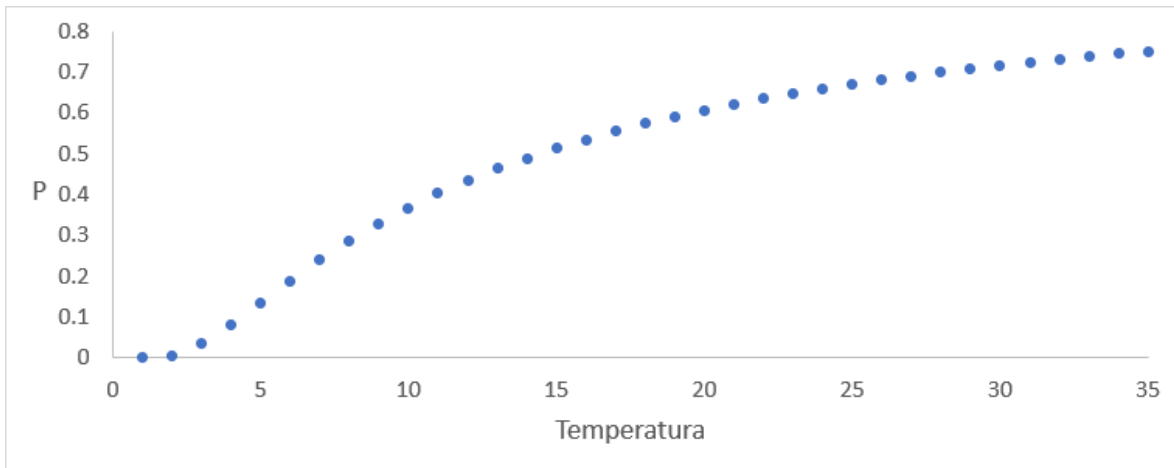


Figura 5: Comportamiento de  $P$  en función de la temperatura.

Fuente: Elaboración propia.

El mecanismo que controla la temperatura del sistema es llamado scheduling de temperatura. El scheduling utilizado en esta memoria tiene dos mecanismos de regulación:

- *Reinicio de temperatura:* Este mecanismo permite reiniciar la temperatura a un valor determinado para permitir la diversificación de la solución. Este reinicio permite reconstruir la solución actual encontrada a un punto de menor calidad, explorando otra

región del espacio de búsqueda. A la cantidad de iteraciones que transcurren entre dos reinicios consecutivos se le llamará ciclo.

- *Variación de temperatura por iteración:* Este mecanismo permite disminuir la temperatura de manera progresiva. Permitiendo inicialmente una diversificación de soluciones y a medida que disminuye la temperatura, los movimientos aplicados se concentran en un espacio más acotado de la región de soluciones permitiendo una intensificación.

## 2.2. Problemas relacionados

Los problemas de optimización de rutas y recolección están presentes en muchos rubros del trabajo humano, a continuación se revisan algunos de los trabajos relacionados al MCPSB. El MCPSB es una aplicación real del Prize Collection Problem (PCP), que a su vez es una generalización del Vehicle Routing Problem (VRP). En esta sección se detalla el VRP y una serie de problemas relacionados con el este problema de optimización combinatoria. Estas variantes consisten en distintas aplicaciones del VRP a casos reales de diversas industrias. Estos problemas tienen complejas restricciones que deben ser satisfechas para encontrar soluciones factibles. Estas restricciones también pueden ser aplicadas en problemas de recolección de leche que tienen directa relación con el MCPSB. Además, las técnicas de resolución de búsqueda de soluciones que se aplican en estos problemas pueden ayudar al diseño de técnicas que permitan encontrar soluciones de manera eficiente para el MCPSB.

### 2.2.1. Vehicle Routing Problem - VRP

El Vehicle Routing Problem (VRP) es un problema de optimización combinatoria que se encuentra en la categoría NP-duro [Archetti *et al.*, 2011]. Una forma de definir el problema se presenta en [Bodin, 1983] como un conjunto de vehículos ubicados en un deposito central utilizados para visitar localidades geográficamente dispersas con el objetivo de satisfacer las demandas requeridas. Un problema relacionado al VRP es el Traveling Salesman Problem (TSP). En el TSP, un vendedor debe encontrar el circuito óptimo para visitar un conjunto de ciudades, buscando minimizar la distancia recorrida visitando cada ciudad una única vez. El VRP puede ser visto como una variación del TSP, con la diferencia de que puede utilizar más de un circuito para visitar todas las ciudades. En la figura 6 se ilustra esta diferencia, el VRP utiliza más de un circuito para visitar el mismo conjunto de ciudades.

En [Dantzig y Ramser, 1959] por primera vez se presenta y analiza matemáticamente un problema de enrutamiento de vehículos. Aquí, el objetivo es encontrar las mejores rutas para una flota de camiones que distribuye combustible, minimizando el costo total de transporte. Este problema considera una planta terminal en donde se carga el combustible y múltiples estaciones de servicio dispersas geográficamente. Las estaciones de servicio requieren una

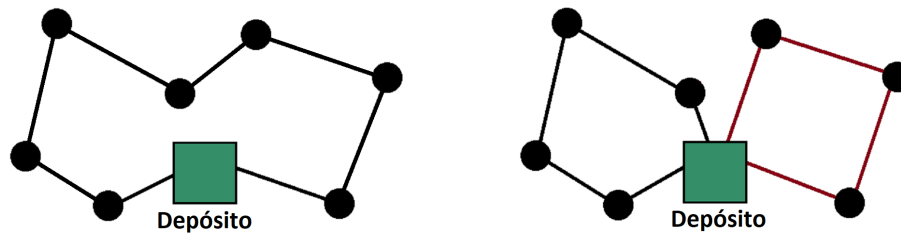


Figura 6: Comparación visual de TSP (izquierda) y VRP (derecha).  
Fuente: Elaboración propia.

cantidad mínima de combustible disponible en todo momento para suplir la demanda de sus propios clientes. La complejidad del problema se debe al gran número de diferentes rutas posibles que existen para conectar todas las estaciones de servicios. Por ejemplo, al considerar un camión que debe visitar 15 estaciones se tienen  $6,53 \cdot 10^{11}$  rutas diferentes que conectan todos los puntos.

En el trabajo mencionado se presenta un modelo de programación lineal entera para el problema, con sus variables y restricciones respectivas que sirvieron de base para trabajos posteriores como [Clarke y Wright, 1964] y [Gillett y Miller, 1974]. Además, se realiza un extenso análisis de posibles variaciones en las restricciones que se aplican a otros problemas de optimización. Por ejemplo, la posibilidad de que exista demanda de múltiples productos distintos, combustible de diferente octanaje, distintos tipos de reciclaje, entre otros. Otra variación que se presenta es considerar que los vehículos de transporte posean distintas capacidades de carga. Finalmente se analiza el caso en que la demanda de cada estación de servicio sea satisfecha con una sola entrega, o en caso contrario, pueda ser satisfecha con múltiples entregas de menor cantidad, lo que permite una relajación de la restricción que permitiría encontrar rutas de menor costo.

Existen diversas variantes del VRP, las cuales buscan resolver problemas de logística reales de recolección y/o entrega, cada uno con distintos contextos y características. En este capítulo se recopilan trabajos en la literatura de variaciones del VRP, que se relacionan con el Milk Collection Problem (MCP).

### 2.3. Prize Collecting Vehicle Routing Problem - PCVRP

Una variante del VRP consiste en la aplicación del Prize Collecting Problem (PCP) a la recolección de bienes. El PCP es presentado en [Balas, 1989] como una variante del TSP. El PCP se describe como el problema de un vendedor que viaja por un conjunto de ciudades, se tiene un costo asociado en viajar entre cada par de ciudades, y en cada ciudad que visita el vendedor recibe una cantidad determinada de beneficio por las ventas que realiza. El objetivo del PCP es maximizar el beneficio obtenido por el vendedor mientras se minimiza el costo de

transporte. Entonces si se aplican los conceptos del PCP al VRP se tiene el PCVRP. Una aplicación del PCVRP es el ya mencionado MCP en donde se debe recolectar la leche necesaria para la operación una planta productora de productos lácteos. Para esto se busca el mejor conjunto de rutas que permiten la recolección de leche desde las granjas productoras, con el objetivo de maximizar el beneficio obtenido y minimizar el costo de transporte asociado.

Un ejemplo real del MCP es el de una compañía productora de lácteos del sur de Chile que necesita leche para elaborar sus productos como fue presentado en el Capítulo 1. En [Montero *et al.*, 2019], se presentan técnicas de búsqueda completa y técnicas metaheurísticas de búsqueda incompleta para resolver instancias del MCP que consideran una gran cantidad de granjas. En primer lugar los autores proponen utilizar un modelo de programación entera. Por otro lado, se presenta una técnica basada en Greedy Randomized Adaptive Process (GRASP) para encontrar soluciones en instancias con hasta 7 rutas.

Para la búsqueda de soluciones con el modelo de programación lineal se utiliza el software CPLEX. Este método permite encontrar soluciones que disminuyen los costos de transporte en un 33 % en comparación a la soluciones que se tenían, que consistían en planificaciones realizadas en base a la intuición y experiencia de los trabajadores. Por otro lado, el enfoque basado en GRASP consiste en tres pasos principales: un pre-procesamiento, una etapa de construcción y una etapa de post-procesamiento. El pre-procesamiento realiza un cálculo de compatibilidades entre granjas para construir conjuntos de granjas cercanas que puedan ser visitadas por un solo camión. En la etapa de construcción se usa un algoritmo similar a una técnica Greedy que considera la leche que falta por recolectar y las distancias recorridas para elegir las granjas a visitar por los camiones. Además, se realiza una selección aleatoria entre las mejores granjas para dar diversidad a las soluciones. El objetivo de la etapa de construcción es encontrar una solución factible inicial utilizando una función de evaluación que minimice el costo de transporte y el costo de la leche recolectada.

Luego de obtener la solución inicial, se procede a la etapa de post-procesamiento que consiste en cuatro movimientos:

- *Reparación*: Se eliminan nodos que puedan mejorar la calidad de la solución en caso de que exista una recolección excesiva de leche.
- *Intensificación*: Se realizan intercambios internos entre las rutas de los camiones para cambiar el orden en el que se visitan las granjas.
- *Intensificación/Diversificación*: Se escogen intercambios para las rutas entre granjas que pertenezcan a rutas de la solución actual y granjas que no han sido consideradas en las rutas actuales.
- *Diversificación*: Se añaden nodos que generen las menores pérdidas de calidad a la solución actual.

La técnica GRASP descrita se ejecuta hasta alcanzar una cierta cantidad de tiempo determinada, y tiene como resultado final la mejor solución encontrada hasta ese momento. En

comparación al método basado en programación lineal entera, la técnica GRASP consigue mejores resultados usando una menor cantidad de recursos cuando se trata de encontrar más de 4 rutas distintas (generando una mejora de hasta un 10.95 % en reducción de costos). Basados en los resultados obtenidos por análisis estadístico, los autores concluyen que los movimientos utilizados en la técnica GRASP permiten explorar de mejor manera grandes espacios de soluciones. En particular, utilizar una mayor cantidad de vehículos y rutas distintas de manera simultanea permite obtener un mejor desempeño.

### **2.3.1. Vehicle Routing Problem with Pick-up and Deliveries - PDVRP**

Existen muchos problemas reales en donde las localidades que deben ser visitadas pueden estar muy lejanas desde la planta de procesamiento, provocando que el costo de transporte sea superior al beneficio obtenido por la entrega o recolección de bienes. Una forma de disminuir el costo de transporte es la utilización de puntos de acopio. Un punto de acopio es una instalación en donde se acumula producto por parte de los productores, para luego ser retirado por parte de los camiones recolectores. Esto permite a granjas pequeñas de difícil acceso depositar su producción en lugares más cercanos a las rutas principales de los camiones. Esta práctica disminuye el costo total de transporte al distribuir la carga en vehículos más ligeros que tienen un menor gasto energético. El costo de transportar la leche en un vehículo de menor tamaño hacia el punto de acopio permite disminuir hasta un 80 % en comparación con el costo de los camiones utilizados por la planta [Dirección General de Transporte Terrestre, 2014].

En el trabajo de [Paredes-Belmar *et al.*, 2016] se propone una forma de resolver el problema mencionado anteriormente basándose en un caso real del sur de Chile, y se considera el primer trabajo en abordar el MCP con puntos de acopio. Para este caso se tiene una planta de producción de lácteos que compra y recolecta leche desde granjas dispersas en una gran área geográfica. Se utiliza una flota homogénea de 80 camiones con 30.000 litros de capacidad para la recolección en más de 500 granjas productoras. El objetivo es encontrar el correcto número y posicionamiento de puntos de acopio para que las granjas distantes puedan entregar su producción, de tal forma que el vehículo pueda recolecta el total de producción de varias granjas en una sola parada, ahorrando costo y tiempo con el objetivo de aumentar el beneficio obtenido por la empresa. En la figura 7 se muestra en (a) las granjas geográficamente dispersas y en (b) se observan 112 puntos de acopio.

El posicionamiento de los puntos de acopio fue resuelto con un modelo de covering codificado en AMPL usando CPLEX. Una vez obtenido los puntos, estos son tratados como granjas de producción en un modelo matemático que se utiliza para encontrar rutas factibles para la recolección. Para esto se propone una técnica metaheurística basada en Ant Colony Optimization, la que genera un set de rutas factibles para la recolección. Para evaluar la técnica se utilizan subconjuntos de granjas que pueden ser resueltos por un algoritmo branch-and-cut. Se observa que incluso utilizando división en zonas más pequeñas, la técnica metaheurística no puede resolver el problema en una cantidad razonable de tiempo. Además, al comparar el

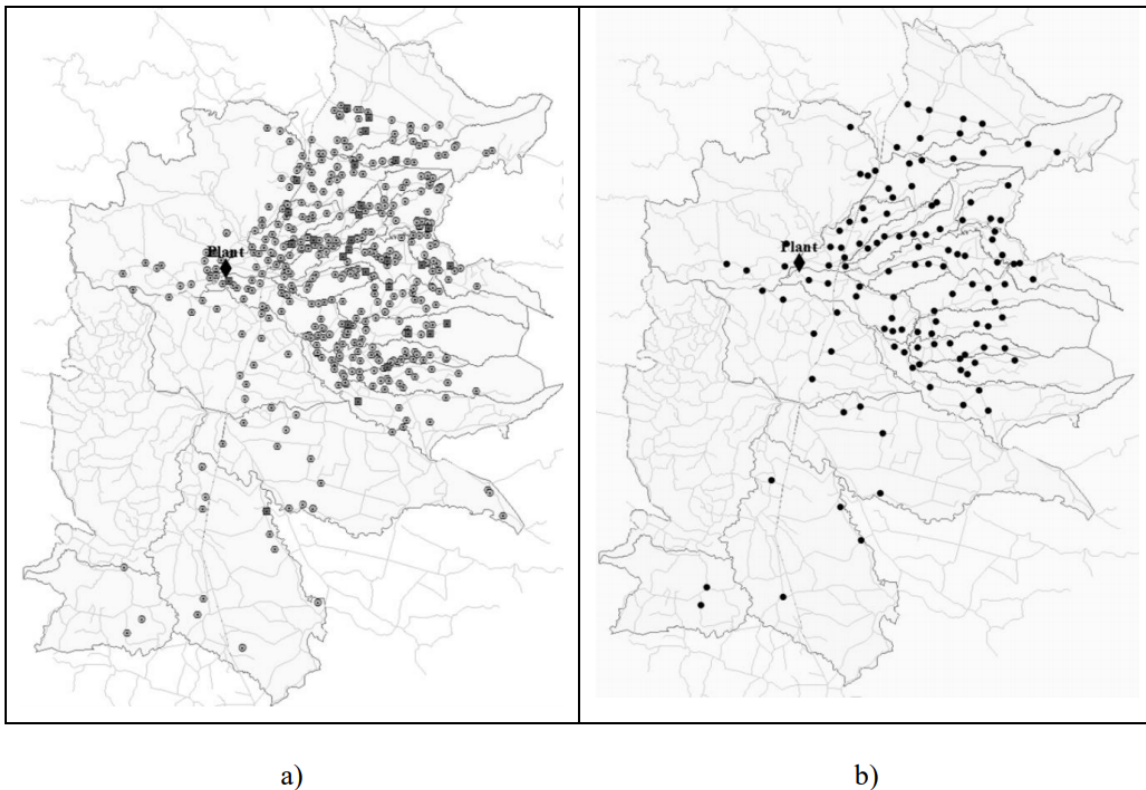


Figura 7: Distribución de granjas (a) y puntos de acopio (b).  
Fuente: [Paredes-Belmar *et al.*, 2016].

beneficio obtenido por la solución sin división de zonas es de un 10.5 % menor que al utilizar divisiones, y tiene un tiempo de ejecución mayor. Sin embargo, la técnica metaheurística es capaz de encontrar soluciones factibles en pocos minutos, lo que es útil cuando se requieren soluciones de manera rápida sin necesidad de ser óptimas. Finalmente, se demuestra experimentalmente que el uso de puntos de acopio reduce el costo de transporte y mejora el beneficio obtenido de manera considerable.

### 2.3.2. Multi Product Vehicle Routing Problem - MPVRP

El Multi Product Vehicle Routing Problem (MPVRP) consiste en el transporte de diversos tipos de bienes. Estos bienes pueden variar en tamaño, temperatura, riesgo, calidad, etc. Si los productos no pueden ser transportados en el mismo vehículo se deben generar distintas rutas para cada producto [Abkowitz y Cheng, 1988]. Por otro lado, existen regulaciones que permiten la mezcla de distintos productos.

Para el caso de recolección de leche, en el MCPSB se pueden requerir distintas calidades de leche para la operación de la planta procesadora. La mezcla de leche es una practica común en distintos países [Incorporated, 2014]. Suponga que se tienen tres leches de calidad de-

creciente: leche de calidad A (que es la más alta), leche de calidad B, y finalmente leche de calidad C. Sin embargo, cuando se mezclan dos o más tipos de leche de distinta calidad, la mezcla resultante se clasifica con la calidad menor de los componentes. Existen casos en que la pérdida por la baja de la calidad en la leche es menor que el costo de transporte necesario para recolectar leche en otra granja.

En la figura 8 se muestra un escenario para ejemplificar el efecto de la mezcla de leche. La figura 8.a representa la distribución de 5 nodos, siendo el nodo 0 la planta procesadora de leche, y los nodos 1 al 4 representan granjas productoras. También se representa el costo de transporte entre cada nodo como un número en los arcos. Finalmente se indica la cantidad de litros de leche producida en cada granja como un número y la calidad de la leche como una letra. Existen tres calidades de leche con distintos beneficios por litro, la leche de calidad A tiene un beneficio de  $1u$ , la leche de calidad B  $0.7u$ , y la leche de calidad C  $0.3u$ .

Se presentan dos conjuntos de rutas, la figura 8.b muestra la mejor ruta posible sin mezclar leche con un beneficio de  $110u$ . Por otro lado, la figura 8.c muestra la mejor ruta permitiendo la mezcla de leche con un beneficio de  $194u$ . Se obtiene un mayor beneficio al mezclar productos de distinta calidad por el ahorro de costo de transporte.

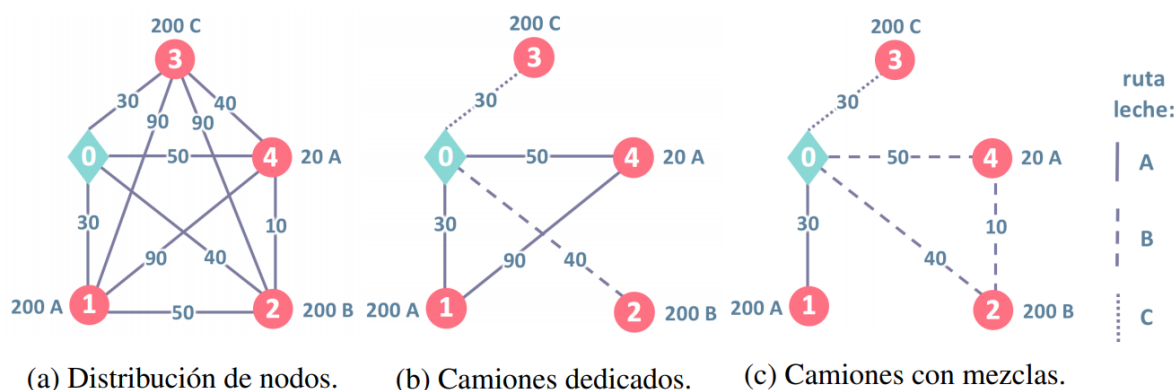


Figura 8: Comparación de rutas sin mezclar calidades (b) y con posibilidad de mezcla (c).  
Fuente: [Soto, 2019].

Permitir la mezcla de leche en un camión de un compartimiento puede permitir encontrar rutas de menor costo para los vehículos. De esta forma, granjas que producen distinta calidad de leche pueden estar cerca geográficamente y el camión es capaz de visitar ambas granjas, recolectando la producción en un mismo compartimiento.

Este problema ha sido abordado en [Paredes-Belmar *et al.*, 2016]. En este trabajo se buscan soluciones al MCP, pero adicionalmente se presenta una demanda de distintas calidades de leches, que se recolectan con una flota de camiones de un solo compartimiento. En el trabajo se presenta la opción de realizar la mezcla de leche en los camiones y también en la planta al entregar la carga. Es importante notar que en este trabajo se considera que todas las granjas productoras deben ser visitadas, sin seleccionar que granjas son visitadas para disminuir costos de transporte.

Los autores proponen un algoritmo basado en programación entera mixta con una técnica basada en branch-and-cut para instancias medianas. Mientras que para el caso de instancias más grandes, se agrupan grandes sectores geográficos de granjas y se resuelven las rutas óptimas con otra técnica basada en branch-and-cut con sectorización para mejorar la eficiencia de la técnica. De los resultados obtenidos, se concluye en primer lugar que los beneficios obtenidos son mayores cuando existe la opción de realizar mezcla de leche de distinta calidad, en comparación a resolver el problema sin utilizar mezcla. Por otro lado, se experimenta con la opción de utilizar camiones con varios compartimentos de tal forma que cada compartimento pueda cargar leche de distinta calidad. Incluso en este caso, la mezcla de leche en los distintos compartimentos permite encontrar soluciones con beneficios mayores en comparación a no permitir la mezcla.

En [Soto, 2019] también se aborda el problema de recolección de leche con selección y mezcla, en este trabajo se propone un algoritmo basado en la metaheurística Simulated Annealing (SA), al que llamaremos Milk Collection Simulated Annealing o MCSA. El algoritmo implementado tiene una fase de construcción enfocada en la producción y cercanía de los nodos. Que permite generar soluciones iniciales factibles de buena calidad. También se diseñan cuatro operadores. Dos operadores de diversificación que permiten añadir nuevos nodos a las rutas generadas y romper las demandas de la planta productora, para realizar búsqueda en regiones infactibles del espacio de soluciones. Finalmente, se presentan dos operadores de intensificación, que reordenan nodos dentro de una misma ruta y eliminan nodos que no sean necesarios para cumplir la demanda de leche, con el objetivo de mejorar la calidad de las soluciones encontradas. Con la técnica MCSA, se alcanzan buenos resultados que permiten encontrar soluciones a grandes instancias de hasta 500 nodos.

### **2.3.3. Multi Compartment Vehicle Routing Problem - MCVRP**

Si los vehículos disponibles tienen distintos compartimientos para cada tipo de producto, se puede generar un único set de rutas para la recolección de todos los tipos de bienes. Esta variación se conoce como Multi Compartment Vehicle Routing Problem (MCVRP).

El Milk Collection Problem with Selection and Blending (MCPSB) utilizando camiones con distintos compartimientos es una aplicación del MCVRP para la recolección de leche. Esto permite separar las calidades de leche en un mismo camión recolector, e incluso realizar mezclas en un mismo compartimento para para ampliar el espacio de soluciones, con el objetivo de encontrar rutas que disminuyan el costo de transporte. En [Caramia y Guerriero, 2010] se aborda el problema de una empresa italiana de productos lácteos, la cual recolecta cuatro tipos distintos de leche en granjas que se ubican en un área rural geográficamente dispersa. Los camiones recolectores poseen múltiples compartimentos que transportan leche sin realizar mezcla de los distintos tipos. Se aborda el problema buscando minimizar la cantidad de camiones utilizados y el largo de las rutas. Se proponen formulaciones matemáticas que son resueltas con el solver CPLEX 7.0 y una técnica de búsqueda incompleta con múltiples reinicios. Las soluciones encontradas muestran una mejora en comparación a las rutas que

eran utilizadas hasta la fecha.

En [Reed *et al.*, 2014], se presenta un problema relacionado al reciclado de desperdicios domésticos en Inglaterra. En este trabajo se menciona que el costo de la logística de la recolección de materiales reciclables representa hasta un 95 % del total. Por otro lado, las regulaciones de desechos sólidos en la Unión Europea imponen presión hacia las autoridades locales para mantener un sistema sustentable [Pires *et al.*, 2011]. Esto genera una creciente importancia en obtener rutas más eficientes y utilizar una menor cantidad de vehículos.

En el trabajo de Reed se utilizan vehículos con hasta cuatro compartimentos para reciclar de forma separada vidrio, papel y otros desechos domésticos recolectados. Las localidades se encuentran geográficamente dispersas y se utiliza una flota de camiones homogéneos en las instancias de los problemas. Para encontrar soluciones, se utiliza una técnica basada en Ant Colony System (ACS). Además se evalúa el uso de clusterización, que consiste en utilizar un algoritmo para generar subconjuntos de localidades cercanas con el objetivo de simplificar el problema al utilizar vehículos que trabajan en un solo conjunto. Las dos etapas principales de la técnica son la construcción de la ruta y la actualización de las feromonas de hormigas.

En la etapa de construcción, las hormigas de forma iterativa construyen rutas individualmente empezando por nodos aleatorios. Estos nodos son una representación de los hogares en donde se recolectan desechos y el deposito en donde se descarga la recolección. En cada paso de la construcción las hormigas aplican una función basada en la distancia, el costo del camino, y la información que otras hormigas dejaron en el camino con los siguientes dos valores:

- La función heurística  $\eta_{ij}$  representa el atractivo del movimiento, usualmente calculado como el inverso de la distancia-costo de ir desde el nodo  $i$  al nodo  $j$ .
- El nivel de feromona en el arco  $(i, j)$  denotado  $\tau_{ij}$  indica qué tan útil ha sido este camino en el pasado.

Con estas dos funciones, la probabilidad con la que la hormiga decide a que nodo  $n$  ir se representa como:

$$p_{in}^k = \frac{(\tau_{in})^\alpha (\eta_{in})^\beta}{\sum_{l \in N_i^k} (\tau_{il})^\alpha (\eta_{il})^\beta}$$

Donde  $N_i^k$  representa el conjunto vecindario factible (i.e los nodos que son directamente accesibles desde el nodo  $i$  y no han sido visitados anteriormente), mientras que  $\alpha$  y  $\beta$  son parámetros de la técnica.

Cuando todas las hormigas han completado una ruta, las hormigas depositan feromonas para comunicar a la colonia sobre caminos interesantes. Por otro lado, las feromonas se evaporan continuamente con el objetivo de dejar de utilizar caminos poco útiles.

En los resultados publicados por Reed, se puede observar que la técnica basada en ACS tiene resultados muy cercanos a las mejores soluciones conocidas. Además, se analiza el resultado de utilizar clusterización en distintas instancias. Se observa que en las instancias de hasta 50 nodos los mejores resultados se obtienen solo con ACS, mientras que en instancias de 100 o más nodos, se obtienen resultados mejores y más rápido con el uso de clusterización. En la figura 9 se muestra visualmente la diferencia entre los dos tipos de soluciones, al utilizar clusters la técnica ACS se utiliza independientemente en pequeños grupos de nodos para crear y mejorar las rutas de recolección.

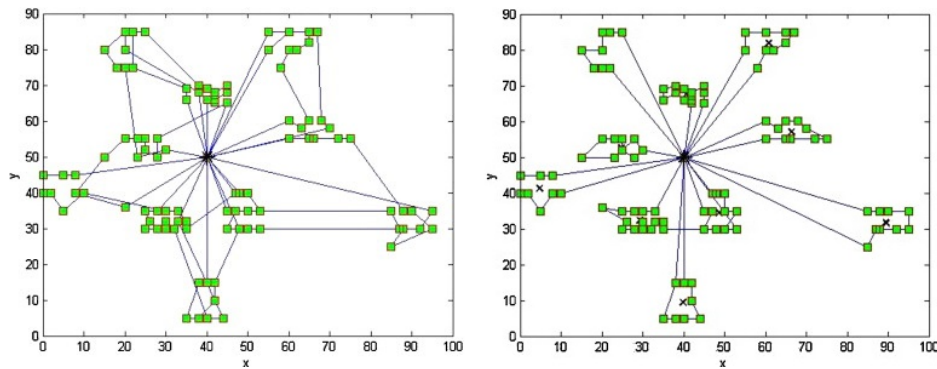


Figura 9: Comparación visual de solución sin clusters (izquierda) y con clusters (derecha).  
Fuente: [Reed *et al.*, 2014]

## 2.4. Time Window Vehicle Routing Problem - TWVRP

En algunos casos, las rutas de un problema deben ser planificadas de forma diferente en distintos instantes de tiempo. Por ejemplo, la recolección de desechos no se realiza todos los días en las mismas rutas, o la recolección de frutas puede variar según la estación del año. En este contexto nace el Time Window Vehicle Routing Problem (TWVRP) que fue propuesto por primera vez por [Dantzig y Ramser, 1959]. En este problema se busca optimizar la recolección diaria de productos desde varias localidades y planificar las rutas en una cierta cantidad de tiempo. Además, se toman en cuenta las variaciones en la producción de las localidades en cada periodo planificado.

Una aplicación del TWVRP se realizó en Quebec para un problema recolección de leche [Dayarian *et al.*, 2015]. En este problema se recolecta leche desde las granjas productoras y se entrega a un conjunto de plantas procesadoras. En un tiempo  $t$ , una planificación es preparada para un horizonte de tiempo  $T$  que consiste en varios conjuntos de días partiendo en  $t + \delta$ . Entonces, la planificación se ejecuta diariamente en el horizonte  $T$ . Las rutas

se diseñan para un conjunto de vehículos que inician su recorrido en distintos depósitos, luego visitan un subconjunto de las localidades (recolectando un solo tipo de producto) y finalmente entregar producto a las plantas. Se busca minimizar el costo de transporte y satisfacer las demandas de la planta. En este problema de recolección cada granja productora es visitada por solo un vehículo y cada vehículo solo visita una granja por día. Para encontrar las rutas de los camiones se implementó un algoritmo de generación de columnas con Branch-and-bound y se utilizó el solver CPLEX con 5 horas de ejecución. Los autores reportan buenos resultados en instancias pequeñas de hasta 20 productores y 5 periodos, mientras que en instancias donde existen cientos de productores no se encontraron soluciones. En este trabajo se menciona que el uso de técnicas metaheurísticas debería ayudar a encontrar soluciones a los problemas más grandes, con cantidades limitadas de tiempos de ejecución.

#### **2.4.1. Síntesis**

Teniendo en cuenta las variantes anteriores, se pueden realizar muchas combinaciones de las diferencias presentadas para obtener un mayor acercamiento a la realidad de la operación de las empresas. Entre ellas destacamos:

- Recolectar toda la producción de sus granjas asociadas o seleccionar las granjas mínimas necesarias para la producción diaria.
- Utilizar camiones con un solo compartimento o usar camiones con múltiples compartimentos (incurriendo en mayores costos).
- Adoptar el uso de puntos de acopio para que las granjas puedan acercar su producción a las principales rutas de recolección.
- Permitir la mezcla en camiones cuando se necesitan distintas calidades de leches.

Por otro lado también se pueden abordar otras problemáticas más complejas. Por ejemplo, considerar la varianza diaria de producción de leche de las granjas utilizando modelos probabilísticos. A continuación se presenta una tabla que resume las distintas variantes de VRP.

#### **2.5. Resumen**

En este capítulo se presenta la base teórica para entender el funcionamiento de técnicas de búsqueda incompleta, y como permiten encontrar soluciones a grandes problemas de optimización. También se presentaron algunas variantes del Vehicle Routing Problem, que tienen distintas aplicaciones en casos reales de diversas industrias, cada una con sus variables y restricciones particulares. Todas estas variaciones estudiadas se pueden aplicar en procesos de

| <b>Variante</b> | <b>Problema a resolver</b>                                                                                                                                                                           |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>VRP</b>      | Se visitan localidades para satisfacer las demandas del problema particular.                                                                                                                         |
| <b>PCVRP</b>    | Se visitan localidades para la recolección de bienes valiosos.                                                                                                                                       |
| <b>PDVRP</b>    | Se visitan localidades y puntos de acopio para recolectar bienes valiosos. Localidades distantes y/o de baja producción acumulan sus bienes en puntos de acopio para disminuir costos de transporte. |
| <b>MPVRP</b>    | Se recolectan bienes diversos utilizando rutas exclusivas para cada tipo o permitir la mezcla de los bienes para reducir costos de transporte.                                                       |
| <b>MCVRP</b>    | Se recolectan bienes diversos utilizando vehículos con varios compartimentos, permitiendo la recolección de distintos bienes en una misma ruta.                                                      |
| <b>TWVRP</b>    | Se visitan localidades para la recolección o entrega de bienes, las rutas se planifican para diversos periodos de tiempo.                                                                            |

Tabla 1: Tabla resumen variantes VRP

recolección de leche, en donde se busca encontrar conjuntos de rutas que permitan optimizar los beneficios obtenidos. Son muchas las técnicas utilizadas para la búsqueda de rutas de recolección, desde técnicas completas como branch-and-cut hasta técnicas metaheurísticas basadas en la naturaleza como Ant Colony Optimización. Por otro lado, se observa que para las variantes más restrictivas y complejas del Milk Collection Problem los trabajos son recientes y existen poca variedad de técnicas aplicadas. Aún queda mucho trabajo por realizar para encontrar mejores resultados que permitan disminuir los recursos utilizados en el transporte de leche.

## CAPÍTULO 3

### PROPUESTA DE SOLUCIÓN

Milk Collection Problem with Selection and Blending (MCPSB) es un problema real de optimización combinatoria, se aplica en casos reales que involucran a muchos camiones recolectores, una gran cantidad de granjas productoras, y distintas calidades de leche. La magnitud y complejidad del problema (NP-duro) no permite la implementación de técnicas de búsqueda completa, que exploran todo el espacio de soluciones y encuentran o estiman los valores óptimos. Por lo tanto, el problema puede ser abordado con otras estrategias y mecanismos de búsqueda, como utilizar algoritmos de búsqueda incompleta.

Se han utilizado técnicas de búsqueda incompleta en [Soto, 2019] y [Paredes-Belmar *et al.*, 2017] para enfrentar el MCPSB, obteniendo soluciones eficientes para instancias pequeñas y medianas, pero teniendo falencias en las instancias de mayor tamaño en términos de los resultados finales y tiempo de ejecución. Según el teorema No Free Lunch que explora la conexión entre algoritmos de optimización [Wolpert y Macready, 1997], se establece que cualquier algoritmo que tenga un excelente rendimiento en una clase de problemas, tendrá un mal rendimiento en otros, es decir, no existe un algoritmo que sea el mejor para todos los problemas e instancias. Por estas razones, en esta memoria se diseñan e implementan técnicas metaheurísticas híbridas que combinan varias metaheurísticas y componentes con objetivos específicos, para realizar una búsqueda eficiente en el espacio de soluciones y enfrentar la dificultad y dimensión del problema.

En este capítulo se define la estructura de representación de las soluciones utilizada, la factibilidad de las soluciones, y un algoritmo de inicialización de soluciones. Luego se presenta el diseño de diversos componentes que son utilizados en distintas versiones de la técnica metaheurística diseñada en esta memoria. Se describen los operadores de diversificación e intensificación que serán utilizados en los algoritmos de búsqueda local. Las técnicas diseñadas se inspiran en tres algoritmos: Greedy, Simulated Annealing (SA) y Hill Climbing First Improvement (HCFI). También se presenta un diseño de dos etapas, que permite combinar SA y HCFI para mejorar la eficiencia de la búsqueda de soluciones. Finalmente se implementan distintos mecanismos de control de parámetros que mejoran la convergencia a soluciones de mayor calidad.

Al final de esta sección, se presentarán 5 algoritmos que se construyen a partir de los componentes presentados. El objetivo es evaluar el efecto de cada componente y entender cómo mejorar el proceso de búsqueda de soluciones para el MCPSB.

### 3.1. Representación de la solución

En este trabajo, las soluciones del MCPSB se representan como el conjunto de rutas que debe recorrer cada camión. La ruta de cada camión representa el set de granjas a visitar y en que orden serán visitadas. Todas las rutas inician y terminan en la planta (nodo 0). Para los siguientes ejemplos, se trabaja con un conjunto total de 30 granjas, donde cada granja es identificada por un número. A continuación se muestra un ejemplo de una ruta para el camión  $t_1$ :

$$\text{Ruta } t_1 = [0, 8, 23, 15, 0]$$

En el ejemplo, para el camión  $t_1$ , su ruta inicia en la planta (nodo 0) para luego viajar al nodo 8, luego viaja al nodo 23 y al nodo 15, recolectando la leche producida por esas granjas, para finalmente volver a la planta y entregar la leche recolectada.

En las instancias que se trabajan en esta memoria, una solución candidata corresponde a rutas de  $n$  camiones, donde cada camión tiene una ruta designada. A continuación se muestra un ejemplo de una solución candidata:

$$\text{Ruta } t_1 = [0, 8, 23, 15, 0]$$

$$\text{Ruta } t_2 = [0, 3, 14, 2, 5, 0]$$

$$\text{Ruta } t_3 = [0, 21, 10, 7, 0]$$

Se observa que los camiones no visitan necesariamente la misma cantidad de granjas, y además, no todas las granjas son visitadas (en este caso, se visitan sólo 10 granjas).

La representación de la solución seleccionada permite controlar la restricción número 4 del modelo matemático presentado en 1.2.3. Esta restricción indica que no pueden existir saltos en el recorrido de los camiones. Es decir, para que una solución sea factible, cada vez que un camión llega a una granja, el recorrido hacia la siguiente debe partir desde la granja visitada. Como la solución se representa como una lista ordenada de nodos a visitar, las rutas de los camiones siempre cumplen esta restricción.

### 3.2. Factibilidad

La factibilidad de las soluciones candidatas en el MCPSB es restringida por el cumplimiento de las restricciones presentadas en el modelo matemático del capítulo 1. Todas las restricciones del problema son controladas durante la ejecución. Por lo tanto, solo se realiza búsqueda en el espacio de soluciones factibles.

### 3.3. Función de evaluación

Para evaluar la calidad de las soluciones que se generan en la técnica propuesta, es necesario definir una función de evaluación. Para esto, se utiliza la función objetivo del modelo matemático presentado en el capítulo 1. La función 2 busca maximizar los beneficios obtenidos por la recolección de leche, el beneficio se compone en dos partes: el valor obtenido por la cantidad de leche recolectada, y por otro lado, el costo del transporte requerido para recorrer las distancias entre cada nodo.

$$fo = \text{Max} \sum_{t \in T} \sum_{q \in Q} V_q z_{tq} - \sum_{t \in T} \sum_{(i,j) \in F} x_{ij}^t C_{ij} \quad (2)$$

**Calidad:** Una solución de alta calidad o calidad mayor se refiere a una solución que tiene un valor alto de la función objetivo al ser evaluada (en términos de un alto valor de la leche recolectada y/o en un bajo costo de transporte). En el caso contrario, cuando el valor de la función objetivo es bajo para una solución particular, se dice que esa solución tiene baja o menor calidad.

### 3.4. Inicialización

Se diseña un algoritmo para construir soluciones iniciales factibles para el MCPSB. Este algoritmo de construcción se basa en Greedy Aleatorizado (muy similar a la fase de construcción de un GRASP). Aquí, se busca satisfacer las demandas mínimas de leche requeridas por la planta procesadora, desde la calidad más alta hasta la calidad de leche más baja. Por cada calidad de leche, se utilizan los camiones disponibles para construir las rutas de forma iterativa. Cuando un camión alcanza su capacidad máxima se continúa con el siguiente camión. Si se alcanza la cantidad mínima requerida de leche de la calidad actual al finalizar el llenado de un camión, se continúa con la siguiente calidad. En este proceso también es posible realizar mezcla de leche en planta. Si se tiene un exceso de leche de calidad alta, esta puede ser utilizada para ser mezclada con leche de calidades más bajas para cumplir con la demanda de todas las calidades. El algoritmo 4 describe el proceso de creación de rutas para todas las calidades de leche.

El punto de partida es una solución vacía, es decir, todos los camiones comienzan desde la planta procesadora y aún no tienen granjas designadas para visitar. Con el objetivo de construir iterativamente una solución inicial factible, se requiere definir una función miope que pueda seleccionar uno de los nodos disponibles, para añadirlo a uno de los camiones de la solución actual. El algoritmo 5 detalla la función miope que permite obtener aleatoriamente uno de los  $n$  mejores nodos, es decir, los  $n$  nodos que tienen la mejor función de evaluación se guardan en una lista, para finalmente escoger uno de los nodos al azar.

---

**Algorithm 4: Greedy**

---

**Result:** Feasible Solution

```
routes  $\leftarrow$  list[nTruckEmptyRoutes];
truck  $\leftarrow$  1;
for  $q \in$  Qualities do
    while !minimunRequired( $q$ ) do
        while !full(routes[truck]) do
            routes[truck].push(miopeFunction( $q$ , truck, routes[truck]));
            if full(routes[truck]) then
                truck ++;
                break;
            end
        end
    end
end
return routes;
```

---

---

**Algorithm 5: Función miope aleatorizada**

---

**Result:** Best Node Randomized

```
nodeValue  $\leftarrow$  list[nNodes];
routecurr  $\leftarrow$  actualTruckRoute;
for node  $\in$  notVisitedNodes(quality) do
    routenew  $\leftarrow$  routecurr.push(node);
    nodeValues.push(eval(routenew));
end
bestN  $\leftarrow$  bestNodes(nodeValues);
noderandom  $\leftarrow$  bestN[random];
return noderandom;
```

---

### 3.5. Operadores de Diversificación

Se diseñan dos operadores que permiten diversificar la búsqueda de soluciones, que tienen el objetivo de modificar la solución actual para escapar de máximos locales y explorar nuevas regiones en el espacio de soluciones. Estos operadores generarán un cambio en el tamaño de las soluciones y se espera que generen cambios sustanciales en términos de la calidad de las mismas.

### 3.5.1. Operador Add

Se implementa el operador Add que permite añadir un nodo que no ha sido visitado anteriormente a la ruta de un camión. Se selecciona un camión aleatorio dentro de la solución actual, y se crea una lista con todos los nodos que se pueden añadir a la ruta sin romper la restricción de capacidad máxima del camión, luego se selecciona de manera aleatoria el nodo que será añadido a la ruta.

El operador Add aplica movimientos a la solución actual que aumentan la cantidad total de nodos que visita un camión. Este operador diversifica la región del espacio de soluciones al añadir nuevas granjas a las rutas de los camiones, las cuales posteriormente pueden ser modificadas o eliminadas por otros operadores. El operador Add también aumenta la cantidad de leche recolectada y el costo de transporte asociado. En la figura 10 se ilustra un movimiento aplicado a una ruta con el operador Add. Para este y los siguientes ejemplos ilustrativos, se considera un beneficio de  $0.4u$  por litro de leche y un costo de  $1u$  por kilómetro recorrido. En la figura se representa la planta como un rombo (PP) y las granjas como cuadrados, en las granjas se muestra su identificador y su producción en litros de leche, por ejemplo el nodo 1 produce 100 litros de leche. Entre los nodos se representa con una flecha el recorrido y la dirección de un camión junto con la distancia recorrida en kilómetros.

En la figura 10.a se muestra el caso de una ruta que inicia desde la planta, para luego visitar el nodo 1 y luego el nodo 2, con un beneficio de  $23u$ . En comparación, en la figura 10.a se añade el nodo 3 a la ruta, y el beneficio aumenta a  $116u$

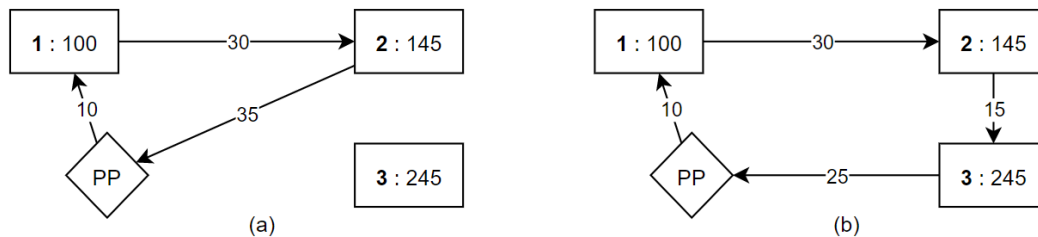


Figura 10: Operador Add antes (a) y después (b) del movimiento aplicado.

Fuente: Elaboración propia.

### 3.5.2. Operador Drop

El operador Drop tiene el efecto contrario al operador Add. Este operador puede eliminar nodos de la ruta de un camión en la solución actual. Se selecciona una camión al azar y se elimina un nodo que pertenezca a la ruta del camión sin romper la restricción de recolección mínima. Este movimiento permite reducir la cantidad de nodos visitados dentro de cada ruta, disminuye la cantidad de leche recolectada y el costo de transporte asociado. Un ejemplo de un movimiento realizado por el operador Drop es pasar del estado de la figura 10.b al estado

de la figura 10.a.

Al disminuir la carga total de leche en los camiones recolectores, este movimiento otorga la posibilidad de añadir nuevos nodos que antes no podían ser considerados por la restricción de máxima capacidad. De esta manera, este movimiento permite escapar de máximos locales y diversificar la búsqueda de soluciones en distintas regiones del espacio de soluciones.

### 3.6. Operadores de Intensificación

Se presentan dos operadores de intensificación, que permiten realizar una búsqueda focalizada en una región de búsqueda más acotada en comparación con los operadores de diversificación. Por lo tanto, el objetivo es realizar pequeñas perturbaciones que permitan explorar una región más acotada del espacio de soluciones y encontrar óptimos locales. Para esto se implementan los siguientes operadores.

#### 3.6.1. Operador Swap Externo

El operador Swap Externo selecciona una ruta aleatoriamente, y permite reemplazar uno de sus nodos por otro nodo que no haya sido visitado por ninguna de las otras rutas. Dentro de la ruta seleccionada, se escoge un nodo de manera aleatoria, y luego se analiza la posibilidad de reemplazar ese nodo por otro nodo que no haya sido visitado. El objetivo del operador es encontrar rutas alternativas que puedan resultar en soluciones de distinta calidad, variando el volumen de recolección o el costo de transporte, sin modificar el largo total de la ruta. Este operador intensifica la búsqueda en una región más acotada, al realizar pequeñas perturbaciones a la solución actual, con el objetivo de encontrar soluciones vecinas alternativas de mayor o menor calidad.

En la figura 10 se ilustra el efecto de este movimiento en una ruta seleccionada. Inicialmente la ruta del camión contemplaba visitar el nodo 1, luego el nodo 2 y finalmente el nodo 3 con un beneficio total de  $106u$  por la recolección de leche. Luego del movimiento la ruta cambia, en donde visita primero el nodo 1 y luego el nodo 2 y finalmente el nodo 4 con un beneficio final de  $110u$ .

Se espera que el operador Swap Externo tenga un efecto menor en la calidad de las soluciones, en comparación al efecto provocado por los operadores de diversificación. La razón principal se refiere a que el reemplazo de un nodo de una ruta modifica en menor medida el volumen de leche recolectado y el costo de transporte asociado.

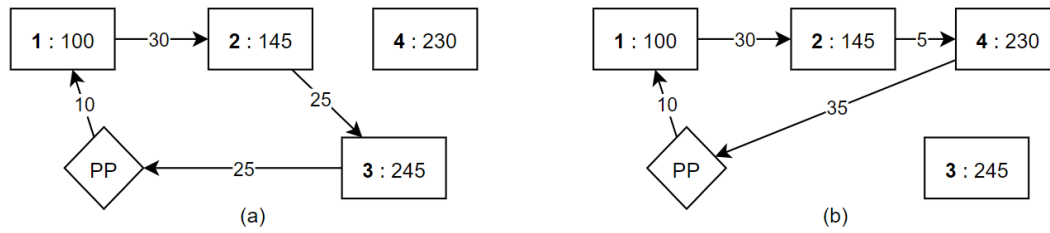


Figura 11: Operador Swap Externo antes (a) y después (b) del movimiento aplicado.  
Fuente: Elaboración propia.

### 3.6.2. Operador Swap Interno

Se implementa también un operador Swap Interno que permite modificar el orden de los nodos visitados dentro de una ruta (en de la solución actual). Por ejemplo, en la figura 12.a el camión comienza su ruta visitando el nodo 1, luego el nodo 2, luego el 4 y finalmente el 3 con un beneficio total de  $183u$ . Luego, al aplicar el movimiento Swap interno, se intercambian los nodos 3 y 4 como se muestra en 12.b. Entonces, la ruta inicia visitando el nodo 1, luego el nodo 2, ahora el nodo 3 y finalmente el nodo 4, lo que resulta en un beneficio de  $188u$ . Este movimiento no modifica la cantidad ni la calidad de leche recolectada, dado que se visitan las mismas granjas. El objetivo de este movimiento es encontrar rutas más eficientes reordenando la forma en que se visitan los nodos para disminuir los costos de transporte. En el ejemplo de la figura 12. La ruta del lado izquierdo tiene un costo de transporte de 105 unidades, y la ruta del lado derecho tiene un costo de 100 unidades. Este operador permite intensificar la búsqueda de soluciones en una región acotada para encontrar soluciones de mayor calidad.

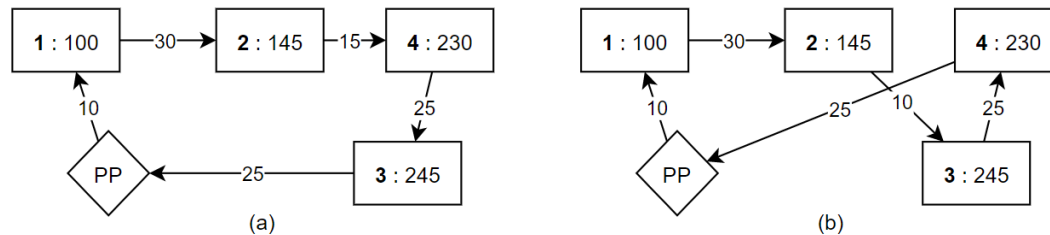


Figura 12: Operador Swap Interno antes (a) y después (b) del movimiento aplicado.  
Fuente: Elaboración propia.

## 3.7. Metaheurísticas Involucradas

En esta sección se definen las metaheurísticas utilizadas en los distintos algoritmos que serán presentados al final del capítulo. Estas metaheurísticas se han implementado anteriormente en distintos trabajos relacionados al MCPSB, realizando búsqueda incompleta con el objetivo de encontrar soluciones a problemas de optimización combinatoria de gran complejidad.

### 3.7.1. Simulated Annealing

Simulated Annealing (SA) será utilizado como una metaheurística de búsqueda iterativa de soluciones que permite escapar de óptimos locales. En la sección 2.1.4 del capítulo anterior, se explica con mayor detalle las principales características de un algoritmo SA. En esta memoria se diseñan e implementan técnicas basadas en el algoritmo SA con distintos operadores y mecanismos de control.

A continuación, en el pseudocódigo 3 se presenta la implementación genérica de SA utilizada en esta memoria. Se realizan  $nCycles$  ciclos de búsqueda, cada uno con  $nIterations$  iteraciones, por lo tanto, en total se realizan  $nCycles * nIterations$  iteraciones durante la ejecución del algoritmo. El objetivo de estos ciclos es controlar la temperatura del sistema  $temp_{curr}$ , donde al inicio de cada ciclo la temperatura se restablece al punto inicial  $temp_0$ . En la siguiente sección se definen otros mecanismos que permiten tener un mayor control del valor de la temperatura.

---

**Algorithm 6:** Implementación de Simulated Annealing

---

**Result:** Best Solution Found

```
 $sol_{curr} \leftarrow InitialSolution();$   
 $sol_{best} \leftarrow sol_{curr};$   
for  $i \leftarrow 0$  to  $nCycles$  do  
     $temp \leftarrow temp_0;$   
    for  $j \leftarrow 0$  to  $nIterations$  do  
         $temp \leftarrow calculateTemperature(j);$   
         $sol_{new} \leftarrow operator(sol_{curr});$   
        if  $eval(sol_{new}) \geq eval(sol_{curr})$  then  
             $sol_{curr} \leftarrow sol_{new};$   
        end  
        else if  $exp(\frac{eval(sol_{curr}) - eval(sol_{new})}{temp}) > rand(0,1)$  then  
             $sol_{curr} \leftarrow sol_{new};$   
        end  
        if  $eval(sol_{new}) \geq eval(sol_{best})$  then  
             $sol_{best} \leftarrow sol_{new};$   
        end  
    end  
end  
return  $sol_{best};$ 
```

---

En cada iteración se busca perturbar la solución actual  $sol_{curr}$  a través de operadores, con el objetivo de encontrar soluciones con un mejor valor de la función de evaluación, o mover la solución actual a una región diferente en el espacio de soluciones factibles. Si la nueva solución  $sol_{new}$  obtenida por uno de los operadores tiene un mejor valor de la función de

evaluación que la solución actual  $sol_{curr}$ , la nueva solución reemplazará la solución actual. En caso contrario, si la función de evaluación tiene un valor menor, el movimiento es aceptado solo si se cumple la condición de empeoramiento, que fue explicada en 2.1.4. Además, se mantiene un registro de la mejor solución  $sol_{best}$  encontrada hasta el momento.

**Scheduling de Temperatura:** Al inicio de cada ciclo, la temperatura del sistema se reinicia hasta un valor  $temp_0$ . Luego, en cada iteración del ciclo, la temperatura del sistema disminuye para modificar la probabilidad de aceptación de soluciones de peor calidad. Cuando la temperatura es alta, aumenta la probabilidad de aceptar un movimiento que resulte en una solución de menor calidad. Por el contrario, si la temperatura es baja, disminuye la probabilidad de aceptar un movimiento que empeore la calidad de la solución.

El objetivo general consiste en asignar una valor alto de temperatura al inicio del ciclo, para aceptar muchos movimientos que modifiquen la solución actual, y mover la solución a una región diferente del espacio de soluciones. Luego, a medida que avanzan las iteraciones dentro de un ciclo, la temperatura disminuye lentamente, para disminuir de manera progresiva la posibilidad de aceptar soluciones de peor calidad, e intensificar la búsqueda en una región más acotada para encontrar soluciones de mayor calidad. Finalmente en las últimas iteraciones del ciclo la temperatura es muy baja, con el propósito de aceptar movimientos que únicamente mejoren la calidad de la solución actual o aceptar empeoramientos leves. En esta memoria se evaluaron 3 scheduling de temperatura distintos: exponencial, lineal y sinusoidal. Se definen las siguientes variables que son utilizadas en los scheduling de temperatura:

- $temp_{curr}$ : Temperatura actual, utilizada en el cálculo de probabilidad de aceptación.
- $temp_0$ : Temperatura inicial, es la temperatura máxima que se utiliza al inicio de cada ciclo.
- $nIterations$ : Cantidad de iteraciones totales dentro de un ciclo.
- $iteration$ : Número de la iteración actual dentro de un ciclo.

**Scheduling exponencial:** Este método de control de temperatura se representa en la ecuación 3. En este caso, la temperatura disminuye rápidamente al inicio, y luego disminuye cada vez más lento durante la ejecución del ciclo, gráficamente se puede observar en la línea roja del gráfico 13. En el eje vertical se representa el valor de la temperatura, y en el eje horizontal se muestra la iteración actual durante la ejecución. La línea roja muestra el cambio de la temperatura durante la ejecución utilizando el scheduling exponencial.

$$temp_{curr} = temp_0 * \alpha^{it} \quad (3)$$

**Scheduling lineal:** Este scheduling busca una variación constante en la temperatura actual del sistema y se representa en la ecuación 4. Visualmente, se puede ver la disminución de la temperatura en la línea verde del gráfico 13.

$$temp_{curr} = temp_0 - (temp_0 * iteration / nIterations) \quad (4)$$

**Scheduling sinusoidal:** También se implementa un scheduling sinusoidal representado en la ecuación 5. El objetivo de este scheduling es mantener una temperatura alta al inicio de cada ciclo (para realizar una mayor diversificación en cada ciclo) y luego disminuir rápidamente la temperatura hasta un punto bajo (donde se pueda explotar una región específica del espacio de soluciones). El decrecimiento de la temperatura se puede ver en la línea púrpura del gráfico 13.

$$temp_{curr} = \frac{temp_0}{2} * \cos(iteration * \frac{\pi}{nIterations}) + \frac{t_0}{2} \quad (5)$$

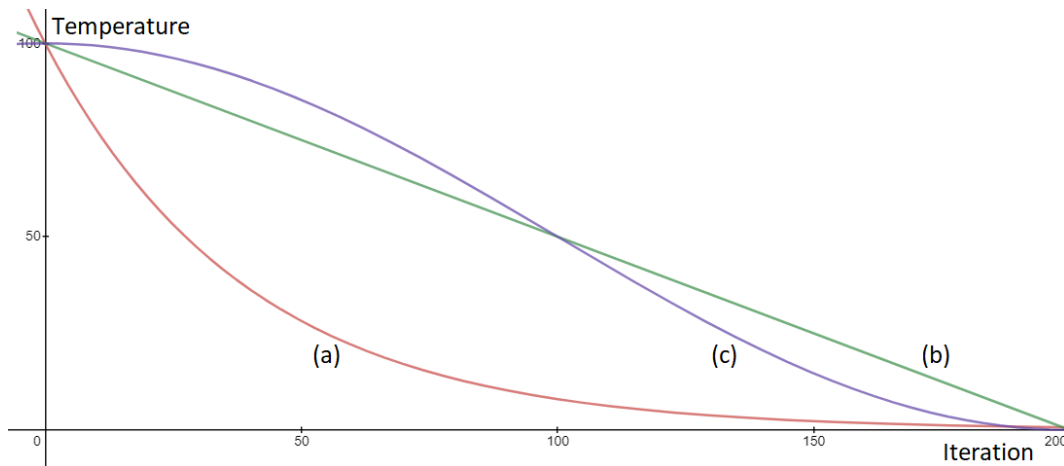


Figura 13: Ejemplo de scheduling exponencial (a), lineal (b) y sinusoidal (c).  
Fuente: Elaboración propia.

### 3.7.2. Hill Climbing First Improvement

Como fue mencionado, se utiliza SA como un método de búsqueda de soluciones que permite escapar de máximos locales. Por otro lado, como un método complementario se utiliza una implementación basada en el algoritmo Hill Climbing First Improvement (HCFI) para encontrar óptimos locales. En el pseudocódigo 7 se detalla la implementación de HCFI utilizada

en esta memoria. En cada iteración del algoritmo se perturba la solución actual a través de operadores que permiten obtener nuevas soluciones factibles. Se observa que a diferencia de la implementación de SA, en este algoritmo no se aceptan movimientos que empeoren la calidad de las soluciones.

---

**Algorithm 7:** Implementación de Hill Climbing First Improvement

---

**Result:** Best Solution Found  
 $sol_{current} \leftarrow \text{InitialSolution}();$   
**for**  $i \leftarrow 0$  **to**  $maxIterations$  **do**  
     $sol_{new} \leftarrow \text{operator}(sol_{curr});$   
    **if**  $\text{eval}(sol_{new}) > \text{eval}(sol_{curr})$  **then**  
         $sol_{curr} > sol_{new};$   
        **if**  $\text{eval}(sol_{curr}) > \text{eval}(sol_{best})$  **then**  
             $sol_{best} \leftarrow sol_{curr};$   
        **end**  
    **end**  
**end**  
**return**  $sol_{best};$

---

Este algoritmo se ejecuta hasta alcanzar el criterio de término, para esta memoria se utiliza un criterio basado en cantidad de iteraciones  $maxIterations$ . El valor de la cantidad máxima de iteraciones varía durante la ejecución de la técnica, y depende de otros parámetros implementados en técnicas posteriores.

### 3.8. Variantes Propuestas

Esta sección presenta cinco metaheurísticas diseñadas de manera incremental para encontrar soluciones al MCPSB. En cada una de las versiones se experimenta con distintas formas de realizar la búsqueda de soluciones y se implementan nuevos mecanismos de control. Además, se presentan los desafíos enfrentados y las mejoras con respecto a la versión anterior.

#### 3.8.1. Greedy Aleatorizado (G-A)

Este acercamiento utiliza un algoritmo basado en Greedy, pero con una selección aleatoria del mejor nodo añadido en cada iteración. Este algoritmo permite construir una solución de manera simple y rápida, pero no utiliza heurísticas sofisticadas durante la ejecución para encontrar las mejores soluciones, y se detiene al obtener la primera solución factible, sin importar el valor de la función objetivo.

### 3.8.2. Simple Simulated Annealing - SSA

Una vez obtenida una solución inicial factible utilizando el algoritmo G-A, se implementan técnicas metaheurísticas para realizar una búsqueda eficiente en el espacio de soluciones factibles. Para esto, se diseña una técnica metaheurística inspirada en SA que permite modificar la solución actual iterativamente, utilizando como base la implementación presentada en 3.7.1.

En esta versión se utilizan los operadores Add y Drop presentados en las secciones 3.5.1 y 3.5.2 respectivamente. Estos operadores permiten añadir y eliminar nodos para obtener nuevas soluciones factibles. Se utiliza un scheduling de temperatura exponencial, para tener una temperatura alta al inicio de cada ciclo y aceptar varios movimientos que modifiquen la solución actual entregada por el ciclo anterior, con el objetivo explorar otras regiones del espacio de soluciones. Luego se disminuye rápidamente la temperatura y se aceptan principalmente movimientos que mejoran la solución actual.

A esta implementación se le denomina Simple Simulated Annealing (SSA), dado que no utiliza operadores ni mecanismos de control sofisticados. Durante la ejecución del algoritmo SSA, los operadores Add y Drop se aplican a la solución actual de manera aleatoria, la probabilidad de aplicar uno de los dos operadores se define con los parámetros *addP* y *dropP*.

### 3.8.3. Two Phases Simulated Annealing - 2SA

Los operadores Add y Drop provocan grandes variaciones en la calidad de las soluciones. Por ende, en las iteraciones finales de cada ciclo donde la temperatura es más baja, es muy difícil aceptar movimientos que empeoren la calidad de la solución. Para esto se incluyen los operadores Swap Externo y el operador Swap Interno, presentados en la sección 3.6.1 y 3.6.2 respectivamente. El operador Swap Externo permite intercambiar un nodo presente en la ruta de un camión, por otro nodo que no este presente en la ruta. Mientras que, el operador Swap Interno permite intercambiar nodos presentes en la misma ruta, buscando disminuir el costo de transporte reordenando la ruta de recolección. Estos operadores permiten realizar movimientos de menor impacto en la búsqueda de soluciones, e intensificar la búsqueda de soluciones en una región más acotada cuando baja la temperatura del sistema.

La variante aquí presentada se llama Two Phases Simulated Annealing (2SA), la cual utiliza los operadores: Add, Drop, Swap Externo y Swap Interno. Sin embargo, la utilización de estos operadores presenta un nuevo desafío. La función de probabilidad de aceptación presentada en la ecuación 1 depende de la diferencia del valor entre la solución actual y la nueva solución obtenida a partir de la aplicación del operador. En el caso de los operadores Add y Drop, esta diferencia es mayor, dado que se añaden o eliminan nodos de la solución. Mientras que, en Swap Externo y Swap interno la diferencia es más pequeña, por que solo se intercambian nodos.

Para enfrentar el problema de la intensidad de las modificaciones de los distintos operadores. En el algoritmo 2SA se implementan dos etapas: etapa de Diversificación y etapa de Intensificación. La etapa de Diversificación tiene los operadores Add y Drop. Por otra parte, la etapa de intensificación utiliza los operadores Swap Externo y Swap Interno. La etapa de Diversificación permite modificar en mayor medida la solución actual y explorar distintas regiones del espacio de soluciones. Por otro lado, la etapa de Intensificación permite realizar pequeñas modificaciones a la solución actual, buscando explotar una región acotada del espacio de soluciones.

Para realizar el cambio de etapa de intensificación a diversificación y viceversa se utiliza una cantidad fija de iteraciones. Para modificar la solución actual entregada por el ciclo anterior, siempre se inicia el ciclo con la etapa de diversificación, luego cambia a la etapa de intensificación, y luego vuelve a diversificación otra vez, y así sucesivamente. Finalmente se termina el ciclo en la etapa de intensificación, para explotar la última región del espacio de soluciones encontrada y tratar de obtener soluciones de mayor calidad.

#### 3.8.4. Improved Two Phases Simulated Annealing - I2SA

La variante 2SA solo realiza el cambio de etapa considerando una cantidad fija de iteraciones, no considera (1) el estado actual de la búsqueda y posibles estancamientos en óptimos locales, (2) obliga a que ambas fases tengan la misma duración y (3) la fase de diversificación inicial podría no ser suficiente para encontrar regiones interesantes. Esta sección presenta una variante llamada Improved Two Phases Simulated Annealing (I2SA) donde se mejoran las condiciones de cambio entre etapas.

El primer problema se presenta cuando la etapa de intensificación se encuentra con máximos locales y la búsqueda se queda atrapada en una pequeña región. Para resolver esto, se diseña un mecanismo que permite pasar a la siguiente etapa al no encontrar mejores soluciones. Se implementa el parámetro llamado *kImprovements*, que intenta detectar posibles estancamientos y permite cambiar de etapa cuando no existen suficientes mejoras en la solución actual.

El segundo problema es que no es necesario realizar la misma cantidad de iteraciones en cada una de las etapas para que la búsqueda sea eficiente. Para enfrentar este desafío se implementa una forma de realizar una cantidad mayor de iteraciones de alguna de las dos etapas con el parámetro *intensificationRatio*. Esto permite asignar más recursos a una de las dos etapas de búsqueda.

Sobre el tercer problema, es necesario que al inicio del ciclo de búsqueda se mueva la solución actual a región diferente del espacio de soluciones, entonces se necesita que la etapa de exploración sea más extensa. Esto se puede lograr implementando un retraso en el primer cambio de diversificación a intensificación, controlado por el parámetro *intensificationDelay*, lo que permite realizar mayores iteraciones en la primera etapa de

diversificación con una alta temperatura, lo que permite la aplicación de muchos movimientos a la solución actual.

Finalmente, se modifica el scheduling de temperatura por un scheduling lineal, para tener una disminución constante de temperatura y llegar a un valor igual cercano a 0 de temperatura en las últimas ejecuciones.

### 3.8.5. Two Phases Routing Algorithm - 2PRA

En la versión I2SA, la etapa de intensificación permite aceptar nuevas soluciones que pueden empeorar la calidad de la solución, por lo que no se concentran los recursos en obtener óptimos locales. Esta sección presenta una variante llamada Two Phases Routing Algorithm (2PRA). Se propone que cada vez que la etapa de diversificación encuentre una nueva región para explotar, el algoritmo realice una búsqueda exclusiva de soluciones vecinas de mayor calidad durante la etapa de intensificación. Para esto, se implementa un algoritmo inspirado en Hill Climbing para reemplazar la etapa de intensificación basada en SA, se mantienen los operadores Swap externo y Swap Interno en este nuevo algoritmo. Esto permite que cuando se estanca la búsqueda de soluciones y no se encuentran mejores soluciones vecinas, el parámetro  $kImprovements$  se encarga de cambiar la búsqueda a la etapa de diversificación para encontrar otra región en el espacio de soluciones.

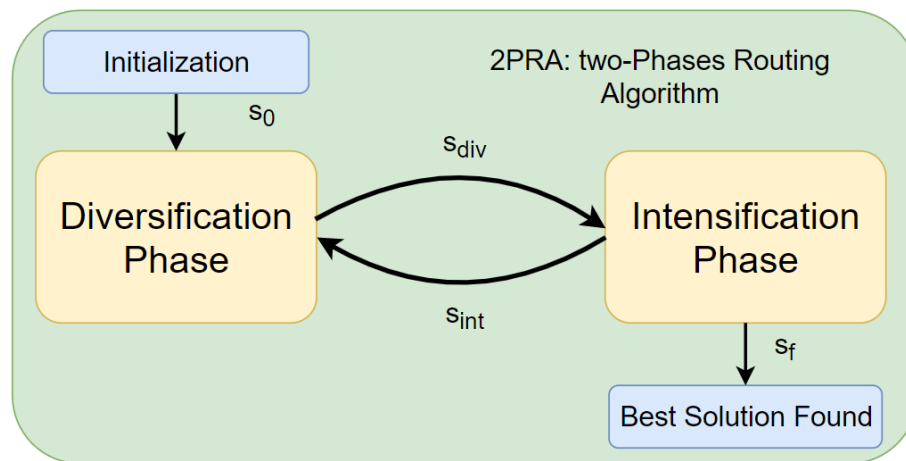


Figura 14: Estructura general de técnica 2PRA. Fuente: Elaboración propia.

Esta técnica se compone de una etapa de construcción de la solución inicial y en dos etapas de búsqueda local. En la figura 14 se ilustra la estructura general.

La ejecución del algoritmo 2PRA inicia con una etapa de construcción. La cual crea una solución factible inicial para el problema con un algoritmo basado en Greedy. Una vez obtenida la solución inicial  $s_0$ , se ejecutan una cantidad determinada de ciclos. Al inicio de cada ciclo, comienza la búsqueda de soluciones en la etapa de diversificación basada en SA. Luego,

cuando se satisface el criterio de cambio de etapa, la solución actual  $s_{div}$  pasa a una etapa de intensificación basada en HCFI. Cuando se satisface el criterio de cambio de etapa, la solución actual  $s_{int}$  vuelve a la etapa de diversificación. Este intercalado de etapas ocurre varias veces durante la ejecución de cada ciclo del algoritmo.

Cuando se satisface el criterio de termino de ciclo, la temperatura actual utilizada en la fase de diversificación se restablece hasta el valor  $temp_0$  y la búsqueda se reinicia en la etapa de diversificación. En todo momento se tiene registro de la mejor solución encontrada  $s_f$ , es decir, la solución con mejor función de evaluación. Cuando se acaba la última iteración del último ciclo, se retorna la solución  $s_f$ .

### 3.9. Cambio de etapas

Dentro de las variantes diseñadas, ciertos algoritmos cuentan con cambios de etapas entre intensificación y diversificación. Esta sección presenta varios mecanismos que permiten realizar estos cambios de etapas.

Dentro de un ciclo de ejecución, se modifica constantemente la solución actual  $sol_{curr}$ . Esta solución se ve afectada por los operadores que se aplican sobre ella, dependiendo si se encuentra en la etapa de diversificación o de intensificación. En la implementación siempre inicia con una etapa de diversificación, esto tiene el objetivo de tomar la solución final del ciclo anterior, y utilizar la etapa de diversificación con una temperatura alta  $temp_0$  para aceptar muchos movimientos sobre la  $sol_{curr}$ . Esto permite mover la  $sol_{curr}$  a una región totalmente diferente del espacio de soluciones.

Se implementa un mecanismo para cambiar la etapa actual llamado scheduling de etapas. Se experimenta con varios métodos para el scheduling de etapas durante la ejecución del algoritmo.

#### 3.9.1. Cantidad fija de iteraciones

La forma más simple de realizar el cambio de etapa, es fijar una cantidad de iteraciones dentro de un ciclo para pasar a la siguiente etapa. Por ejemplo, si un ciclo tiene 100 iteraciones ( $nIterations = 100$ ), la cantidad fija podría ser 10. En este caso, el ciclo comienza con una etapa de diversificación, luego en la iteración 10 pasaría a la etapa de intensificación, luego en la iteración 20 pasaría a la etapa de diversificación nuevamente, y así sucesivamente se intercalan las etapas.

### 3.9.2. Cantidad de iteraciones sin mejora

Otro método para el scheduling de etapas es utilizar información durante la ejecución del algoritmo para aplicar el cambio. En este caso, se utiliza un contador, que lleva registro de la cantidad de iteraciones en las cuales no hubo una mejora en la solución actual. La cantidad límite de iteraciones sin mejora es un parámetro del algoritmo llamado *kImprovements*. Como ejemplo, si *kImprovements* es igual a 5, y en cualquiera de las dos etapas, se realizan 5 iteraciones en donde no hubo un movimiento que mejora la calidad de la solución actual, el algoritmo cambiaría de etapa.

### 3.9.3. Cantidad de iteraciones desequilibradas

Este método añade un componente adicional a cualquiera de los dos scheduling recién mencionados. El objetivo de este scheduling es priorizar una de las dos etapas. Por ejemplo, podría ser útil tener una mayor cantidad de iteraciones en la etapa de intensificación. Si modificamos el scheduling Cantidad fija de iteraciones, podríamos tener dos límites diferentes, uno para la etapa de diversificación con un valor de 10, y uno para la etapa de intensificación con un valor de 20, entonces se realizarían el doble de iteraciones totales en las etapas de intensificación. Este desequilibrio entre las etapas es controlado por un parámetro llamado *intensificationRatio*.

## 3.10. Conclusiones

En este capítulo fueron presentados diversos operadores, componentes y mecanismos de control, que se utilizan en varios algoritmos de búsqueda local diseñados para resolver el Milk Collection Problem with Selection and Blending. Se presentan ilustraciones de los operadores que permiten realizar modificaciones a las soluciones encontradas. Se describen a modo general varios algoritmos de búsqueda local que se construyen de manera incremental, en los que se utilizan distintos componentes y parámetros durante la ejecución de estas técnicas, para mejorar la calidad de las soluciones encontradas y la eficiencia del proceso de búsqueda.

En primer lugar se presenta un algoritmo para construir soluciones iniciales basado en el algoritmo Greedy, para construir soluciones factibles iniciales de manera rápida, y que luego sean utilizadas en otros algoritmos de búsqueda más sofisticados.

Se diseña un algoritmo metaheurístico de búsqueda de soluciones basado en Simulated Annealing (SA). Este algoritmo denominado Simple Simulated Annealing (SSA), tiene dos operadores llamados Add y Drop que permiten aplicar perturbaciones a la soluciones para realizar una búsqueda eficiente en el gran espacio de soluciones del problema.

Se implementa un nuevo algoritmo llamado Two Phases Simulated Annealing, que presenta dos operadores de intercambio de nodos: Swap Interno y Swap Externo. Estos operadores permiten realizar perturbaciones de menor intensidad a las soluciones con el objetivo de explotar soluciones en regiones más acotadas. Para mejorar la eficiencia de la búsqueda, se separan los operadores en dos etapas: una etapa de diversificación con los operadores Add y Drop que permiten explorar distintas regiones del espacio de búsqueda. Y por otro lado, una etapa de intensificación que utiliza los operadores Swap Interno y Swap Externo que permiten explotar regiones más acotadas y encontrar óptimos locales.

Luego se mejoran distintos aspectos del algoritmo 2SA con una nueva variante llamada Improved Two Phases Simulated Annealing o I2SA. En la cual se implementan distintos mecanismos de control para el cambio de etapas, que permiten mejorar la relación y el trabajo en conjunto de las etapas de diversificación e intensificación, y se mejora la eficiencia de la búsqueda para ahorrar recursos computacionales. En esta versión también se implementan nuevos mecanismos de control de temperatura para la etapa de diversificación basada en SA.

Finalmente, se implementa un algoritmo llamado Two Phases Routing Algorithm o 2PRA. Que mejora el algoritmo I2SA al reemplazar la etapa de intensificación basada en SA, por una etapa de intensificación basada en Hill Climbing, la cual no permite aceptar nuevas soluciones de peor calidad durante la etapa de intensificación. Este cambio busca mejorar el trabajo en conjunto de los mecanismos de diversificación e intensificación, y también mejorar la eficiencia del proceso de búsqueda de soluciones.

## CAPÍTULO 4

### VALIDACIÓN DE LA SOLUCIÓN

En este capítulo se presenta el método experimental y los resultados obtenidos en la implementación de las técnicas metaheurísticas presentadas en el capítulo 3. Se especifican las características del entorno de ejecución, la sintonización de parámetros y las instancias utilizadas para las pruebas. Luego, se analizan los resultados y el efecto de los componentes utilizados en las distintas técnicas. Se presentan los mejores resultados del valor de las soluciones encontradas, y los tiempos de ejecución involucrados en la ejecución de las pruebas. Finalmente, se realiza una comparación de los mejores resultados, con valores de trabajos anteriores realizados sobre el Milk Collection Problem with Selection and Blending (MCPSB).

#### 4.1. Instancias

Con el objetivo de evaluar las técnicas propuestas, se trabaja con dos conjuntos de instancias. El primer conjunto, llamado instancias de prueba, consiste en instancias pequeñas de 40 nodos. Por otro lado, se tiene un conjunto de instancias, denominadas instancia reales, que consisten en escenarios de hasta 500 nodos. Para ambos conjuntos de instancias, el costo de transporte por kilómetro recorrido corresponde a 1 unidad ( $u$ ). Mientras que, se tienen tres calidades de leche, denominadas a, b y c, con un beneficio de  $0.003u$ ,  $0.021u$  y  $0.009u$  respectivamente. A continuación se explica con mayor detalle las instancias utilizadas.

##### 4.1.1. Instancias reales

En el conjunto de instancias reales, se experimenta con una instancia de producciones y distancias reales, que consiste en un conjunto de 500 granjas reales dispersas en un área geográfica del sur de Chile, que fue presentada en [Paredes-Belmar *et al.*, 2017]. Además se tienen 5 subconjuntos de esta instancia, contruidos por [Soto, 2019] utilizando una clus-terización de los nodos con el algoritmo K-means. Es decir, se agruparon las granjas más cercanas en 5 subconjuntos. El objetivo de utilizar estas instancias es para realizar pruebas con instancias de distintos tamaños, que permitan analizar el rendimiento y la eficiencia de la técnica en diversos escenarios.

En la tabla 2 para cada instancia se muestra la cantidad de nodos, y la cantidad mínima requerida de litros de leche ( $L$  de aquí en adelante) para cada calidad. Se observa que para cada instancia, existe una instancia con la misma cantidad de nodos (instancia.0). Estos pares de instancias se componen de los mismos nodos, pero en las instancias.0 la planta no requiere una cantidad mínima de leche para su operación. Por lo tanto, se elimina una restricción para encontrar soluciones al problema.

| Instancia | Nodos | Camiones | Qa    | Qb     | Qc    |
|-----------|-------|----------|-------|--------|-------|
| r1        | 128   | 9        | 35000 | 195000 | 15000 |
| r1.0      | 128   | 9        | 0     | 0      | 0     |
| r2        | 143   | 9        | 20000 | 200000 | 15000 |
| r2.0      | 143   | 9        | 0     | 0      | 0     |
| r3        | 142   | 9        | 25000 | 190000 | 20000 |
| r3.0      | 142   | 9        | 0     | 0      | 0     |
| r4        | 31    | 3        | 8997  | 41658  | 8634  |
| r4.0      | 31    | 3        | 0     | 0      | 0     |
| r5        | 56    | 6        | 15000 | 140000 | 10000 |
| r5.0      | 56    | 6        | 0     | 0      | 0     |
| r500      | 500   | 35       | 80000 | 800000 | 75000 |
| r500.0    | 500   | 35       | 0     | 0      | 0     |

Tabla 2: Detalle de instancias reales

Es importante mencionar, que la suma de las instancias de la r1 a la r5, no resulta en un equivalente a la instancia de r500 nodos. Esto se debe a que se asignó de manera aleatoria la calidad de la leche en las instancias creadas por el algoritmo K-means, con el objetivo de tener una cantidad similar de nodos productores de cada calidad de leche. Por ejemplo, en la instancia 1 de 128 nodos, existen aproximadamente 43 nodos productores de leche de calidad A, 43 nodos de calidad B y 43 nodos de calidad C.

Cada nodo de las instancias produce una cantidad de litros determinada de una sola calidad de leche, existen nodos con un gran volumen de producción de hasta 25000  $L$ , mientras que otros nodos producen menos de 100  $L$  de leche.

En todas las instancias reales, la capacidad de los camiones es de 30000  $L$ . Mientras que la cantidad de camiones disponibles para cada instancia depende de la cantidad total de leche a recolectar. La capacidad total de los camiones recolectores, es al menos un 10 % superior al volumen total de leche requerida por la planta. Por ejemplo, la cantidad total de litros requerida para la instancia r1 es de 2450000  $L$ , mientras que la capacidad total de los camiones es de 2700000  $L$ , que equivale a un 10.2 % de exceso en capacidad.

#### 4.1.2. Instancias de prueba

Se experimenta con un conjunto de 6 instancias pequeñas con una menor cantidad de nodos. Estas instancias fueron creadas por [Soto, 2019] a partir de un subconjunto de 40 nodos, que fueron extraídos del conjunto de 500 nodos reales mencionado anteriormente. Estas instancias pequeñas se utilizan para realizar pruebas a las distintas versiones de la técnica, ya que al poseer una menor cantidad de nodos las pruebas pueden ser realizadas en una cantidad menor de tiempo, lo que es necesario dada la complejidad de sintonizar parámetros en una técnica metaheurística.

| Instancia | Camiones | Capacidad | Qa    | Qb    | Qc    | Holgura |
|-----------|----------|-----------|-------|-------|-------|---------|
| t1        | 7        | 20000     | 60000 | 40000 | 10000 | 21 %    |
| t2        | 5        | 30000     | 60000 | 40000 | 10000 | 27 %    |
| t3        | 4        | 30000     | 0     | 0     | 0     | 100 %   |
| t4        | 5        | 30000     | 0     | 0     | 0     | 100 %   |
| t5        | 5        | 30000     | 10000 | 70000 | 10000 | 40 %    |
| t6        | 5        | 20000     | 40000 | 40000 | 10000 | 10 %    |

Tabla 3: Detalle de instancias de prueba

En la tabla 3 se muestran las características de las instancias de prueba construidas. Para cada instancia se muestra la cantidad de camiones recolectores disponibles, la capacidad máxima de los camiones en litros, y la cantidad de litros requerida para cada calidad de leche.

Las seis instancias de prueba se basan en un conjunto de 40 nodos. No tienen diferencias en las posiciones geográficas, cantidades de producción y calidad de leche. Sin embargo, cada instancia difiere en las restricciones del problema de optimización, como la cantidad de camiones disponibles y su capacidad máxima de recolección, y en la cantidad mínima de leche requerida para cada calidad requerida por la planta. La última columna de la tabla 3 representa el exceso de capacidad de recolección en comparación al mínimo requerido. Por ejemplo, para la instancia t1, la capacidad total de los camiones es de 140000L, mientras que el total de la leche requerida es de 110000L, lo que otorga un 21 % de holgura para recolectar más leche de la mínima requerida.

En el set de instancias de prueba se observa que existen instancias como t3 y t4 que no requieren un mínimo de leche, por lo tanto, tienen un 100 % de holgura, mientras que la instancia t6, apenas tiene un 10 % de holgura para recolectar leche.

## 4.2. Entorno Experimental

Todos los experimentos ejecutados fueron realizados con 20 semillas diferentes, para disminuir el efecto de la naturaleza estocástica de los algoritmos diseñados. Sobre el hardware utilizado, todas las pruebas fueron realizadas en un computador con procesador Intel(R) Core i3-10100 4,30 GHz, 24 GB de RAM DDR4, en una distribución Ubuntu x64 20.04.

## 4.3. Parámetros

Se realizan experimentos preliminares con el objetivo de comparar los distintos algoritmos propuestos. Estas técnicas tienen parámetros que modifican el comportamiento de la ejecución del algoritmo. Se tienen parámetros numéricos continuos y discretos. A continuación

se presentan los parámetros utilizados en las técnicas y una breve explicación de su funcionamiento.

1. *addP*: Probabilidad de aplicar operador Add en etapa de diversificación.
2. *swapExp*: Probabilidad de aplicar operador Swap Externo en etapa de intensificación.
3. *temp<sub>0</sub>*: Temperatura máxima del sistema, asignada en cada inicio de ciclo.
4. *nCicles*: Cantidad total de ciclos de búsqueda.
5. *nIterations*: Cantidad de iteraciones en cada ciclo.
6. *kImprovement*: Limite de iteraciones sin mejoras a la solución actual, permite cambiar a la siguiente etapa.
7. *intensificationRatio*: Permite asignar más o menos iteraciones a la etapa de intensificación.
8. *intensificationDelay*: Permite retrasar el primer cambio a la etapa de intensificación al iniciar el ciclo.

#### 4.4. Sintonización

La elección correcta del valor de cada parámetro es un problema de optimización adicional. Para comparar el rendimiento de las distintas versiones implementadas, se utiliza la misma cantidad de recursos computacionales y se mantienen constantes algunos parámetros compartidos en distintas versiones. Se realizó una sintonización manual de los parámetros utilizados en la comparación de las distintas versiones. Para esto, se utilizó como conjunto de prueba las instancias de prueba para ajustar los parámetros, y se utilizaron las instancias reales como conjunto de validación. Los valores de los parámetros resultantes se muestran en la tabla 4.

| Parámetro                   | Valor  |
|-----------------------------|--------|
| <i>addP</i>                 | 0.5    |
| <i>swapExp</i>              | 0.5    |
| <i>temp<sub>0</sub></i>     | 35     |
| <i>nCicles</i>              | 50     |
| <i>nIterations</i>          | 100000 |
| <i>kImprovement</i>         | 0.01   |
| <i>intensificationRatio</i> | 0.5    |
| <i>intensificationDelay</i> | 0.05   |

Tabla 4: Parámetros utilizados para la comparación de las distintas versiones

| Instancia | G-A  | SSA   | 2SA         | I2SA        | 2PRA         |
|-----------|------|-------|-------------|-------------|--------------|
| t1        | -325 | 492   | <b>675</b>  | <b>675</b>  | <b>675</b>   |
| t2        | -105 | 732   | 865         | 873         | <b>898</b>   |
| t3        | 0    | 1166  | 1197        | <b>1239</b> | 1237         |
| t4        | 0    | 1208  | 1272        | <b>1296</b> | <b>1296</b>  |
| t5        | -433 | 1177  | 1233        | <b>1241</b> | <b>1241</b>  |
| t6        | -380 | 547   | <b>604</b>  | <b>604</b>  | <b>604</b>   |
| r1        | 2332 | 4759  | <b>4926</b> | 4916        | 4921         |
| r2        | 1031 | 3628  | 3825        | <b>3857</b> | 3824         |
| r3        | 1016 | 3308  | 3495        | <b>3557</b> | 3494         |
| r4        | 350  | 1238  | <b>1251</b> | <b>1251</b> | <b>1251</b>  |
| r5        | 2007 | 2891  | <b>3006</b> | 2999        | 2993         |
| r500      | 9511 | 16132 | 17545       | 17491       | <b>17566</b> |
| r1.0      | 0    | 5958  | <b>5999</b> | 5966        | 5993         |
| r2.0      | 0    | 4476  | 4523        | <b>4645</b> | 4593         |
| r3.0      | 0    | 4068  | 4180        | <b>4210</b> | 4182         |
| r4.0      | 0    | 1451  | <b>1464</b> | <b>1464</b> | <b>1464</b>  |
| r5.0      | 0    | 1451  | 3317        | <b>3350</b> | 3327         |
| r500.0    | 0    | 19057 | 20176       | 20386       | <b>20430</b> |

Tabla 5: Resultados de las técnicas implementadas

#### 4.5. Resultados de las versiones

A continuación se muestran los resultados obtenidos por cada versión de la técnica implementada, también se presentan los problemas, los desafíos enfrentados y las decisiones tomadas luego del análisis de los resultados obtenidos. El código fuente de cada versión se encuentra en el siguiente repositorio online<sup>2</sup>. La calidad de los resultados obtenidos se evalúan con la función de evaluación presentada en la ecuación 2. La función de evaluación busca maximizar el beneficio obtenido por la empresa, considerando el costo de recolectar la leche y el valor obtenido por la leche recolectada. Aquí, un valor alto, representa un beneficio económico para la empresa, mientras que un valor negativo resulta en una pérdida para la empresa, y no resulta rentable realizar la de recolección.

En primer lugar se analiza el algoritmo Greedy Aleatorizado (G-A), que fue presentado en la sección 3.8.1. Los resultados de G-A para las todas las instancias se muestran en la columna Greedy de la tabla 5, en donde se muestra el mejor valor obtenido en 20 semillas para cada una de las instancias.

Con el algoritmo G-A se obtiene una solución inicial factible para todas las instancias. Sin embargo, la calidad de estas soluciones es pobre. Además, se aprecia que en varias instancias

<sup>2</sup>[https://github.com/matiasmoreno/MCPSPB\\_SA](https://github.com/matiasmoreno/MCPSPB_SA)

se tienen valores negativos, donde los costos de recolección superan a los beneficios obtenidos. Mientras que en otras instancias se obtienen valores iguales a 0. Esto se debe a que estas instancias no presentan un mínimo de recolección, por lo que el algoritmo G-A entrega una solución vacía, que es factible en la definición del problema matemático.

Una vez obtenida una solución inicial, se procede a realizar una búsqueda de soluciones de mayor valor en el espacio de soluciones factibles. Se implementa una técnica metaheurística que permita realizar búsqueda local utilizando el algoritmo Simulated Annealing. Esta versión se denomina Simple Simulated Annealing (SSA) que fue presentada en 3.8.2.

En la columna SSA de la tabla 5 se muestran los mejores valores de la función objetivo con respecto a G-A para las 20 semillas en cada una de las instancias. Se observa que con la inclusión de los operadores Add y Drop se obtiene una mejora en la función objetivo en todas las instancias y todos los valores de las soluciones son mayores a 0. Al comparar las instancias reales, con sus pares sin restricción de mínima recolección, los resultados son mejores cuando no es necesario cumplir con una cuota mínima de recolección por cada calidad de leche. Esto se produce por una relajación de las restricciones del problema, lo que permite encontrar más movimientos factibles para aplicar a las soluciones, que no serían factibles con la restricción de recolección mínima.

La tercera columna muestra los resultados de 2 Phases Simulated Annealing (2SA) que fue presentada en 3.8.3. Los resultados de la ejecución muestran que los valores de la función objetivo son mejores en todas las instancias en comparación con el algoritmo SSA. La utilización de dos etapas y la separación de diversificación e intensificación permiten explorar distintas regiones del espacio de soluciones y explotar las regiones más prometedoras.

La cuarta columna de la tabla muestra los resultados de Improved 2 Phases Simulated Annealing (I2SA) descrito en 3.8.4. En los resultados se observan mejoras en todas las instancias de prueba t1 a t6. En el caso de las instancias reales, las instancias más beneficiadas de esos cambios, con las instancias r1, r2, r3 y r500 que corresponden a las instancias con mayor cantidad de nodos. Esto se explica por una mejora en la eficiencia del algoritmo, al cambiar más rápido a la siguiente etapa si no se encuentran mejores soluciones.

Finalmente, la última columna muestra los resultados de 2 Phases Routing Algorithm (2PRA) que utiliza Simulated Annealing para la etapa de diversificación y Hill Climbing First Improvement (HCFI) para la etapa de intensificación. De los resultados se observa que se encuentran algunos nuevos máximos en comparación con la técnica anterior, especialmente en las instancias de 500 nodos. Esto se explica por una nueva mejora en la eficiencia del algoritmo, al no aceptar soluciones de peor calidad en la etapa de intensificación, permitiendo que el parámetro de control *kImprovements* realice el cambio de etapa más rápido. De esta forma, se realizan más iteraciones de búsqueda en otras regiones del espacio de soluciones.

## 4.6. Comparación con Técnicas Existentes

Hasta este punto, todas las pruebas de las distintas técnicas se realizan con una cantidad limitada de recursos y con los mismos parámetros. Para realizar un análisis más exhaustivo de la técnica, se muestran a continuación los resultados finales de las pruebas sobre la técnica 2PRA con más recursos computacionales y utilizando parámetros sintonizados específicamente para esta versión. Se trabaja con esta versión porque es la que obtiene mejores resultados en las instancias mas grandes trabajadas.

Para la configuración de parámetros de la técnica 2PRA se realiza una sintonización automatizada, utilizando el sintonizador Evoca [Riff y Montero, 2013] con las instancias reales. Los valores de los parámetros resultantes se muestran en la tabla 6.

| Parámetro                   | Valor  |
|-----------------------------|--------|
| <i>addP</i>                 | 0.5    |
| <i>swapExP</i>              | 0.5    |
| <i>temp<sub>0</sub></i>     | 35     |
| <i>nCicles</i>              | 500    |
| <i>nIterations</i>          | 500000 |
| <i>kImprovement</i>         | 0.01   |
| <i>intensificationRatio</i> | 0.5    |
| <i>intensificationDelay</i> | 0.05   |

Tabla 6: Parámetros utilizados en los experimentos finales

Se utilizarán como referencia dos trabajos realizados sobre el MCPSB. En primer lugar se estudian los resultados obtenidos por [Paredes-Belmar *et al.*, 2016], el que utiliza una técnica de búsqueda completa basado en el algoritmo CPLEX. Mientras que, para realizar comparaciones sobre las instancias más grandes trabajadas, se analizan los resultados obtenidos por [Soto, 2019], quien utiliza un algoritmo metaheurístico basado en Simulated Annealing, al que llamaremos Milk Collection Simulated Annealing (MCSA). Ambos trabajos fueron mencionados con mayor detalle en la sección 2.3.2.

En primer lugar, se analizan los resultados obtenidos a partir de los experimentos realizados en las instancias de prueba. La tabla 7 muestra los mejores resultados obtenidos por el algoritmo CPLEX, el algoritmo MCSA y el algoritmo propuesto 2PRA en las columnas Best. También se muestra el valor promedio (Avg) encontrado por el algoritmo 2PRA utilizando 20 semillas. Y finalmente en las columnas Gap CPLEX y Gap MCSA se realiza una comparación porcentual entre los valores encontrados por las técnicas en comparación. La diferencia porcentual o Gap se calcula con la ecuación 6, en donde CPLEX se refiere al mejor resultado encontrado por el algoritmo CPLEX, y 2PRA se refiere al mejor resultado encontrado por 2PRA. Para la comparación con MCSA solo se reemplaza los valores de CPLEX por el mejor valor encontrado por MCSA.

| Instancia | <b>CPLEX</b> | <b>MCSA</b> | <b>2PRA</b> |      |           |          |
|-----------|--------------|-------------|-------------|------|-----------|----------|
|           | Best         | Best        | Best        | Avg  | Gap CPLEX | Gap MCSA |
| t1        | <b>750</b>   | 717         | 675         | 484  | 10.0      | 5.8      |
| t2        | <b>969</b>   | 948         | 877         | 746  | 9.5       | 7.5      |
| t3        | <b>1254</b>  | 1249        | 1239        | 1231 | 1.2       | 0.8      |
| t4        | <b>1311</b>  | 1230        | 1296        | 1295 | 1.2       | -5.4     |
| t5        | <b>1241</b>  | 1003        | <b>1241</b> | 1234 | 0.0       | -23.7    |
| t6        | <b>604</b>   | <b>604</b>  | <b>604</b>  | 595  | 0.0       | 0.1      |

Tabla 7: Mejores resultados CPLEX vs MCSA vs 2PRA en instancias de prueba, Gap de mejores resultados

$$Gap = \frac{CPLEX - 2PRA}{CPLEX} * 100 \quad (6)$$

Si el valor de Gap es positivo, quiere decir que el resultado presentado tuvo un rendimiento menor al resultado del trabajo anterior. Por otro lado, si Gap es negativo, quiere decir que el algoritmo propuesto obtiene mejores resultados para esa instancia. Finalmente, si el valor de Gap es 0, quiere decir que ambas soluciones llegan al mismo resultado.

Se observa que en las instancias t1 y t2 la técnica 2PRA se acerca al resultado obtenido por CPLEX, pero con una diferencia cercana a un 10 %, mientras que en las instancias t3 y t4, la diferencia es de solo un 1.2 %. Finalmente en las instancias t5 y t6 se encuentra el mejor valor conocido. Esto se puede explicar por las características de las instancias t1 y t2, las cuales son dos de las instancias más restrictivas, es decir, requieren de una cantidad mayor de leche de alta calidad, mientras que la holgura de la capacidad de los camiones es pequeña. Por otro lado, para las demás instancias, la holgura de la capacidad es mayor, y se requiere una cantidad menor de leche de calidad más alta.

En cuanto a la comparación con el algoritmo MCSA, para las instancias t1, t2 y t3 el algoritmo MCSA obtiene mejores resultados. Mientras que para las instancias t4, t5 y t6, el algoritmo propuesto 2PRA obtiene mejores resultados, con hasta un 23.7 % de diferencia.

Uno de los objetivos de esta memoria es obtener resultados de manera eficiente, es decir, utilizar una menor cantidad de recursos computacionales para encontrar las soluciones al problema. En la tabla 8 se observan los tiempos de ejecución promedio en segundos para el algoritmo CPLEX, el algoritmo MCSA y la técnica 2PRA para todas las instancias de prueba. Para el algoritmo 2PRA, la columna Best se refiere al tiempo que toma la técnica propuesta en encontrar la mejor solución, la columna Avg se refiere al promedio del tiempo total de ejecución de cada una de las 20 semillas, y la última columna Std Dvt se refiere a la desviación estándar de los tiempos de ejecución total para cada semilla.

El algoritmo CPLEX fue ejecutado por 30 minutos en cada instancia. Por lo que su tiempo de ejecución es mayor en todas las instancias. La técnica 2PRA tiene tiempos de ejecución pro-

| Instancia | <b>CPLEX</b> | <b>MCSA</b> | <b>2PRA</b> |            |        |
|-----------|--------------|-------------|-------------|------------|--------|
|           | Avg          | Avg         | Best        | Avg        | Std Dv |
| t1        | 1800         | 235         | 171         | <b>193</b> | 11     |
| t2        | 1800         | <b>142</b>  | 143         | 223        | 15     |
| t3        | 1800         | <b>32</b>   | 103         | 278        | 34     |
| t4        | 1800         | <b>88</b>   | 110         | 285        | 39     |
| t5        | 1800         | <b>242</b>  | 224         | 303        | 29     |
| t6        | 1800         | <b>141</b>  | 17          | 188        | 23     |

Tabla 8: Tiempos de ejecución CPLEX vs MCSA vs 2PRA en instancias de prueba

medio menores, cercanos a 4 minutos. En cuanto al algoritmo MCSA, este consigue menores tiempos de ejecución promedio en 5 de las 6 instancias de prueba. Se observa una tendencia del algoritmo propuesto de no tener un muy buen rendimiento en instancias pequeñas y muy restrictivas.

A continuación se analizan los resultados obtenidos con el conjunto de instancias reales que poseen requisitos mínimos de cada calidad de leche. Para esto, se tiene como referencia los resultados encontrados por [Soto, 2019] con el algoritmo MCSA. En la tabla 9, en la columna MCSA Best se muestran los mejores resultados obtenidos para cada instancia por el algoritmo MCSA. Mientras que en las columnas 2PRA se encuentra el mejor resultado obtenido (Best), el resultado promedio (Avg) obtenido con las 20 semillas. Y finalmente la columna Gap muestra la diferencia porcentual entre ambas técnicas, utilizando la ecuación 6.

| Instancia | <b>MCSA</b> | <b>2PRA</b>  |       |      |
|-----------|-------------|--------------|-------|------|
|           | Best        | Best         | Avg   | Gap  |
| r1        | <b>5018</b> | 4966         | 4931  | 1.0  |
| r2        | <b>3884</b> | 3881         | 3846  | 0.1  |
| r3        | 3497        | <b>3578</b>  | 3516  | -2.3 |
| r4        | <b>1307</b> | 1251         | 1251  | 4.2  |
| r5        | 3018        | <b>3019</b>  | 2994  | 0.0  |
| r500      | 16531       | <b>17637</b> | 17504 | -6.7 |

Tabla 9: Mejores resultados MCSA vs 2PRA en instancias reales, Gap de resultados promedio

Se observa que ambas técnicas encuentran distintos máximos en distintas instancias. Ambas técnicas tienen un rendimiento similar en las instancias r1, r2 y r5. MCSA supera a 2PRA en r4 por una diferencia de 4.2 %. Sin embargo, 2PRA obtiene mejores resultados en r3 y en r500, encontrando la mayor diferencia en la instancia de 500 nodos, en donde la técnica propuesta obtiene soluciones con hasta un 6.7 % de diferencia.

A continuación, en la tabla 10 se comparan los tiempos de ejecución en segundos de 2PRA y MCSA. La columna MCSA Avg muestra el tiempo promedio de ejecución de la técnica de MCSA. Mientras que para la técnica propuesta 2PRA se muestra el tiempo de ejecución hasta encontrar la mejor solución (Best), el tiempo de ejecución promedio de las 20

semillas (Avg) y finalmente la desviación estándar de los tiempos de ejecución (Std DV).

| Instancia | MCSA       | 2PRA |            |        |
|-----------|------------|------|------------|--------|
|           | Avg        | Best | Avg        | Std Dv |
| r1        | 1547       | 234  | <b>550</b> | 37     |
| r2        | 1850       | 104  | <b>645</b> | 28     |
| r3        | 2283       | 596  | <b>713</b> | 30     |
| r4        | <b>209</b> | 75   | 288        | 12     |
| r5        | 568        | 189  | <b>347</b> | 13     |
| r500      | 34178      | 465  | <b>489</b> | 19     |

Tabla 10: Tiempos de ejecución MCSA vs 2PRA en instancias reales

Se observa que en las instancias r1, r2, r3, r5, y r500 se requiere de un menor consumo de recursos. La diferencia es más notable en la instancia completa de 500 nodos, en donde el tiempo de ejecución promedio de 2PRA es 70 veces menor que la técnica MCSA.

Por último, se presentan los resultados de los experimentos realizados para las instancias reales, sin un mínimo requerido para cada calidad de leche. En la tabla 11, en la columna MCSA Best se encuentran los valores máximos obtenidos por MCSA. Por otro lado, para el algoritmo propuesto 2PRA se presentan los mejores resultados (Best) y el resultado promedio obtenido por las 20 semillas (Avg), mientras que en la columna Gap se muestra la diferencia porcentual calculada con la ecuación 6.

| Instancia | MCSA        | 2PRA         |       |       |
|-----------|-------------|--------------|-------|-------|
|           | Best        | Best         | Avg   | Gap   |
| r1.0      | 5253        | <b>6041</b>  | 6006  | -15.0 |
| r2.0      | 4095        | <b>4679</b>  | 4635  | -14.3 |
| r3.0      | 3763        | <b>4249</b>  | 4206  | -12.9 |
| r4.0      | <b>1488</b> | 1464         | 1464  | 1.6   |
| r5.0      | 3179        | <b>3350</b>  | 3341  | -5.4  |
| r500.0    | 15938       | <b>20666</b> | 20425 | -29.7 |

Tabla 11: Mejores resultados MCSA vs 2PRA en instancias reales sin mínimo de leche, Gap de resultados promedio

Se observa que en las instancias r1.0, r2.0, r3.0, r5.0 y r500.0 se alcanzan nuevos valores máximos. Los valores promedios también son bastante buenos y cercanos a los valores máximos encontrados. La mayor diferencia se encuentra en la instancia r500.0 donde el nuevo máximo encontrado tiene un valor de hasta un 29.7 % mayor que la técnica MCSA. Se observa una tendencia de que mientras más grande es la instancia, la técnica 2PRA encuentra mejores valores.

Los tiempos de ejecución promedio de la técnica se muestran en la tabla 12. El detalle de las columnas es el mismo que el de la tabla 10.

| Instancia | MCSA       | 2PRA |            |        |
|-----------|------------|------|------------|--------|
|           | Avg        | Best | Avg        | Std Dv |
| r1.0      | 636        | 53   | <b>523</b> | 26     |
| r2.0      | <b>504</b> | 343  | 573        | 28     |
| r3.0      | 679        | 172  | <b>543</b> | 27     |
| r4.0      | <b>44</b>  | 10   | 262        | 5      |
| r5.0      | <b>173</b> | 129  | 358        | 12     |
| r500.0    | 5993       | 132  | <b>441</b> | 22     |

Tabla 12: Tiempos de ejecución MCSA vs 2PRA en instancias reales sin mínimo de leche

Se observa que los tiempos de ejecución son menores en las instancias r1.0, r3.0 y r500.0. La mayor diferencia se observa en las instancias más grandes, en donde se obtienen tiempos hasta 13 veces menores.

Según los resultados obtenidos, el mayor aporte de esta memoria radica en las instancias más grandes. La técnica es capaz de trabajar con instancias de 500 nodos, 35 camiones, 3 calidades de leche alcanzando nuevos valores máximos, con tiempos de ejecución mucho menores.

Para analizar el rendimiento de la técnica con recursos aún más acotados, se realizan diversas ejecuciones en la instancia r500 con distintas cantidades de tiempos de ejecución. A continuación, en la figura 15 en el eje vertical se muestran el mejor valor de la función objetivo encontrada para cada caso, y en el eje horizontal se muestra la cantidad de minutos que la técnica fue ejecutada en una escala logarítmica. La línea azul representa el mejor valor obtenido en la ejecución de 20 semillas, la línea naranja, muestra el valor promedio del mejor valor obtenido, y la línea gris representa el menor valor obtenido en las ejecuciones.

A partir de los resultados, se observa que los resultados de la técnica tienen una relación directa con la cantidad de tiempo que se ejecuta la búsqueda, en la figura 15 se puede ver que los resultados promedios, máximos y mínimos son mayores cuando la técnica se ejecuta por 12 minutos. Por otro lado, en un tiempo mínimo de 0.5 segundos, la técnica igualmente encuentra buenos resultados. Si comparamos el valor promedio de las soluciones en 0.5 segundos de ejecución con las soluciones de 12 minutos de ejecución, la diferencia es de un 5.1 % en el valor de la función objetivo. Esto demuestra que la técnica propuesta es sumamente eficiente al momento de encontrar buenas soluciones con una cantidad muy limitada de recursos.

#### 4.7. Convergencia y análisis de scheduling de Temperatura

En esta sección se analiza el comportamiento del valor de la función objetivo de la solución actual  $sol_{curr}$ , según la temperatura del sistema y los cambios de etapas a medida que avanza la ejecución de los algoritmos presentados. Este análisis nos permite observar posibles

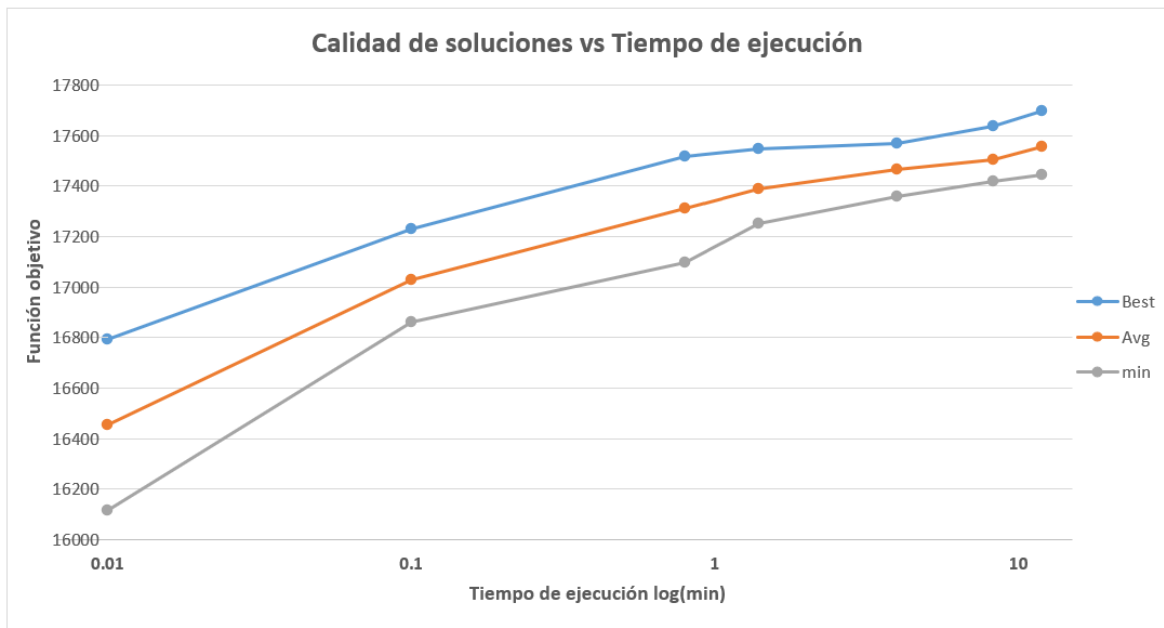


Figura 15: Resultados de instancia r500 en diferentes tiempos de ejecución.  
Fuente: Elaboración propia.

problemas y mejoras en distintos momentos de la búsqueda de soluciones.

**SSA:** En la figura 16 se observa el efecto de la disminución de la temperatura en la calidad de la solución actual. En eje vertical izquierdo representa el valor de la función de evaluación de la solución actual. El eje vertical derecho representa el valor de la temperatura actual. El eje horizontal representa el número de la iteración actual. La línea roja representa la temperatura en cada iteración, la línea púrpura representa el valor de la función de evaluación. Los valores utilizados para generar la figura 16 (y las siguientes figuras), fueron obtenidos con la ejecución de los algoritmos en la instancia r1. En la figura se representan dos ciclos de búsqueda. Se puede ver que hay una relación inversa entre la temperatura y el valor de la solución actual. Al inicio, cuando la temperatura es alta, se aplican muchos movimientos a la solución actual, y el valor de la función objetivo baja rápidamente. Luego, cuando la temperatura desciende, el valor de la solución actual aumenta progresivamente hasta un punto en donde no encuentra otras mejoras a la solución actual.

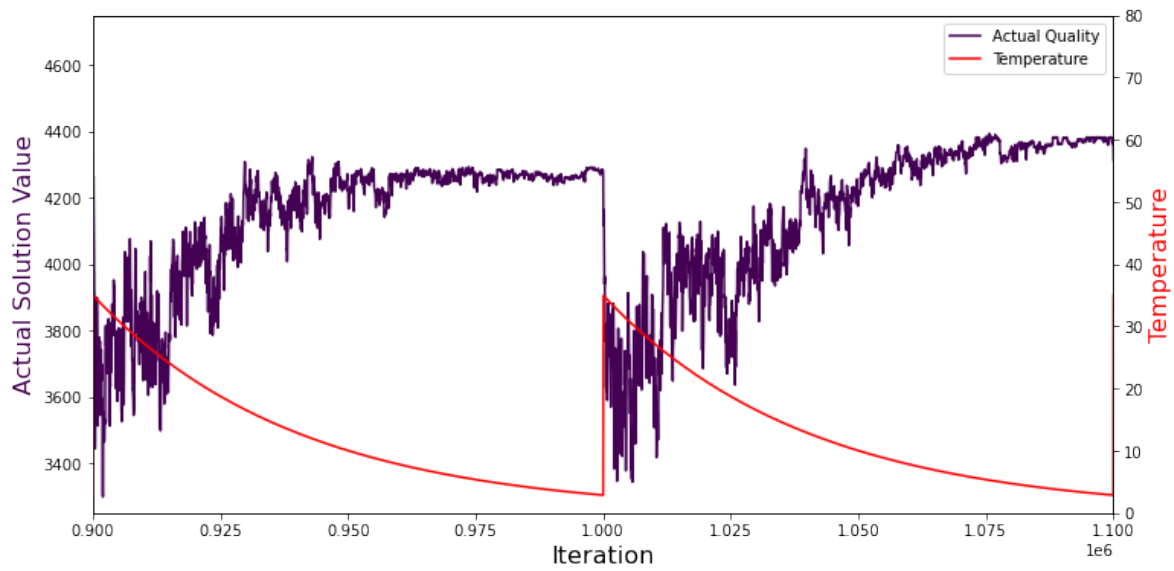


Figura 16: Valor de solución actual durante dos ciclos en la ejecución de SSA.  
Fuente: Elaboración propia.

**2SA:** En la figura 17 se observa el efecto del cambio de etapa en la calidad de la solución actual. En eje vertical izquierdo representa el valor de la función de evaluación de la solución actual. El primer eje vertical derecho representa el valor de la temperatura actual. El segundo eje vertical derecho nos indica si la iteración actual está en etapa de diversificación o intensificación. El eje horizontal representa el número de la iteración actual. La línea roja representa la temperatura en cada iteración, la línea púrpura representa el valor de la solución actual en cada iteración, por último, cuando la línea azul está en el valor 0 del eje derecho significa que la iteración está en la etapa de diversificación, y el valor 1 representa cuando la iteración está en la etapa de intensificación.

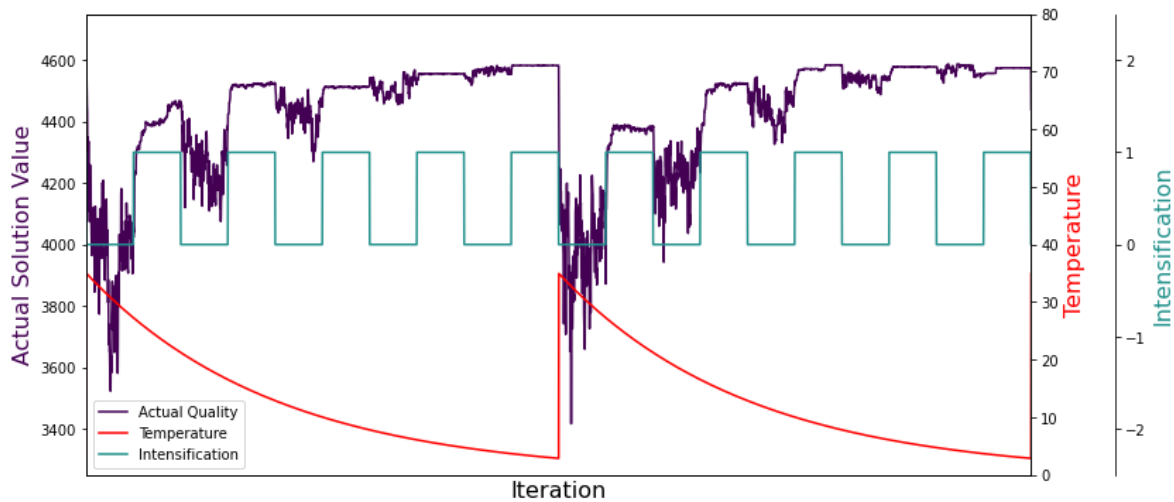


Figura 17: Valor de solución actual durante dos ciclos en la ejecución de 2SA.

Fuente: Elaboración propia.

Al inicio del ciclo, la temperatura tiene un valor alto y desciende rápidamente, luego desciende con menor velocidad durante el ciclo. Por otro lado, al inicio del ciclo se ejecuta la etapa de diversificación. Estos dos factores, la alta temperatura y la etapa de diversificación provocan grandes cambios en la solución actual, la que desciende y asciende rápidamente en su función de evaluación, con el objetivo de mover la solución actual a otra región del espacio de soluciones. Luego, cuando la temperatura disminuye y ocurre el intercambio de etapas, el valor de la solución actual comienza a ascender nuevamente. El algoritmo es efectivo en encontrar buenos resultados, se puede ver que en cada cambio de la etapa de diversificación a la etapa de intensificación hay un rápido aumento en el valor de la solución actual, lo que se explica por la intensa explotación de las soluciones vecinas en una región de soluciones más acotada.

**I2SA:** Se observa en la figura 17 que en las últimas etapas de intensificación, la calidad de la solución tiene poco movimiento, no hay mejoras ni empeoramientos. Una explicación a esto, es que las iteraciones, están repitiendo los mismos movimientos varias veces, hasta que llega la siguiente etapa de exploración y permite mover la solución actual a una región diferente. En la figura 18 se observa el comportamiento de los componentes implementados sobre la calidad de la solución actual, durante la ejecución de dos ciclos, en la misma instancia y el mismo intervalo que las pruebas anteriores.

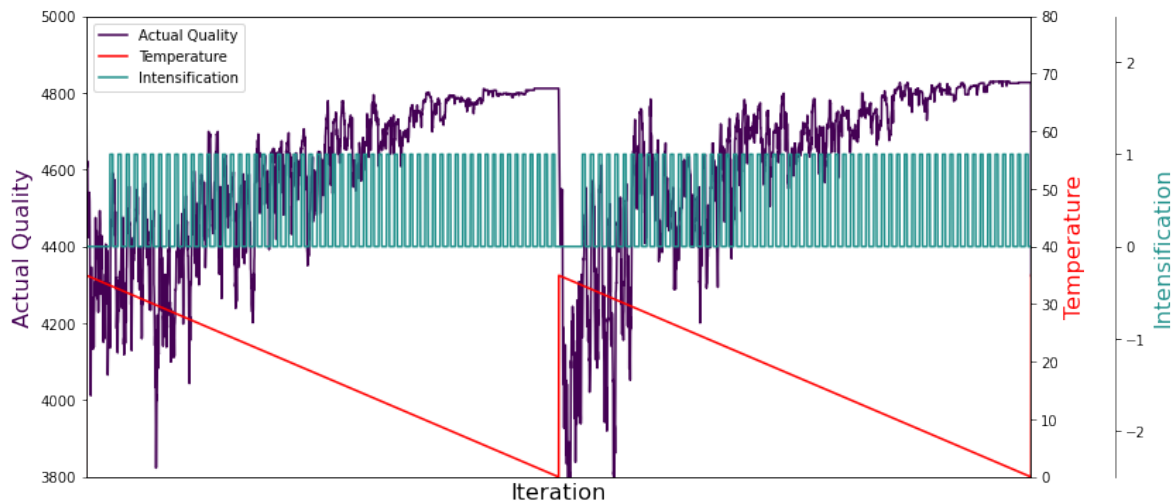


Figura 18: Valor de solución actual durante dos ciclos en la ejecución de I2SA.

Fuente: Elaboración propia.

En la figura 18 se observa que la aplicación de *kImprovements* permite realizar más cambios de etapas cuando son necesarios, es decir, cuando la solución actual no encuentra mejores soluciones vecinas. También se observa que la aplicación de *intensificationRatio* hace que las etapas de intensificación sean más cortas que las etapas de diversificación. Por otro lado, *intensificationDelay* permite tener una primera etapa de diversificación más extensa, para asegurar el cambio de la solución actual a otra región del espacio de soluciones. Finalmente el scheduling de temperatura lineal, permite realizar una intensificación más concentrada en las ultimas iteraciones del ciclo. Se observa como en las ultimas iteraciones no se realizan muchos movimientos que permitan empeorar la calidad de la solución.

**2PRA:** En la figura 19 se observa el comportamiento de la búsqueda de soluciones. La variación en el valor de la solución actual es similar al de la técnica SA3, dado que la principal diferencia radica en no aceptar movimientos que empeoren la solución durante la etapa de intensificación.

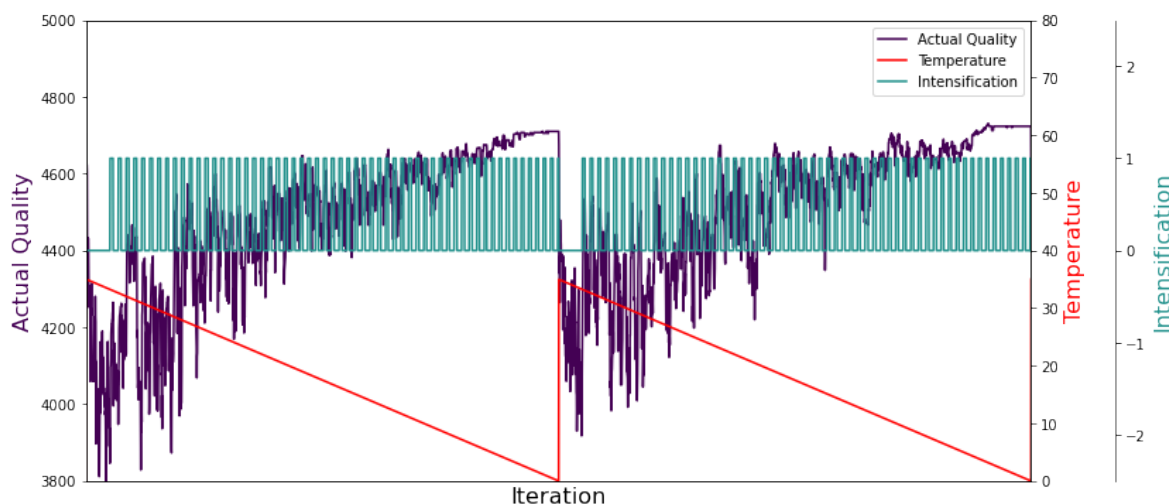


Figura 19: Valor de solución actual durante dos ciclos en la ejecución de 2PRA.  
Fuente: Elaboración propia.

#### 4.8. Efecto de las distintas etapas en las rutas de recolección

Cada etapa de la técnica tiene un objetivo distinto en el proceso de búsqueda de soluciones para el MCPSP. A continuación se analiza el caso de un conjunto de rutas para la instancia r4, con imágenes que ilustran la evolución las rutas en cada etapa de la técnica y como afectan estas modificaciones el valor de la función objetivo.

**Etapas de construcción:** La primera parte de la técnica consiste en construir una solución inicial factible con el algoritmo Greedy aleatorizado. En la figura 20 se presenta un grafo de nodos que representan las granjas de la instancia r4. Los ejes verticales y horizontales representan la latitud y longitud reales de los nodos, cada nodo es representado por un punto de un color verde, amarillo o rojo, para las calidades A, B y C respectivamente. Al lado de cada nodo se muestra el identificador de la granja y debajo se muestra la cantidad de litros de leche que produce. La planta procesadora se representa como un cuadrado negro, mientras que las rutas se muestran como líneas punteadas, una línea azul para la ruta del camión 1, y una línea naranja para la ruta del camión 2. Entre cada camino que utiliza un camión se muestra el costo asociado de utilizar esa ruta con un número de color gris en la mitad del arco. Finalmente se indica con una flecha el inicio de la ruta desde la planta, y el color de la flecha representa la calidad de leche mezclada resultante de la recolección. En la figura 20 el camión 1 inicia su ruta en la planta, luego viaja en orden a los nodos 31, 4, 22, 16, 19, 1, 13, 25 para finalmente volver a la planta, recolectando 29857 de leche de calidad A. Mientras que el camión 2 visita en orden los nodos 26, 8, 5, 23, 11, 17 y 2, recolectando 29829 litros de calidad B

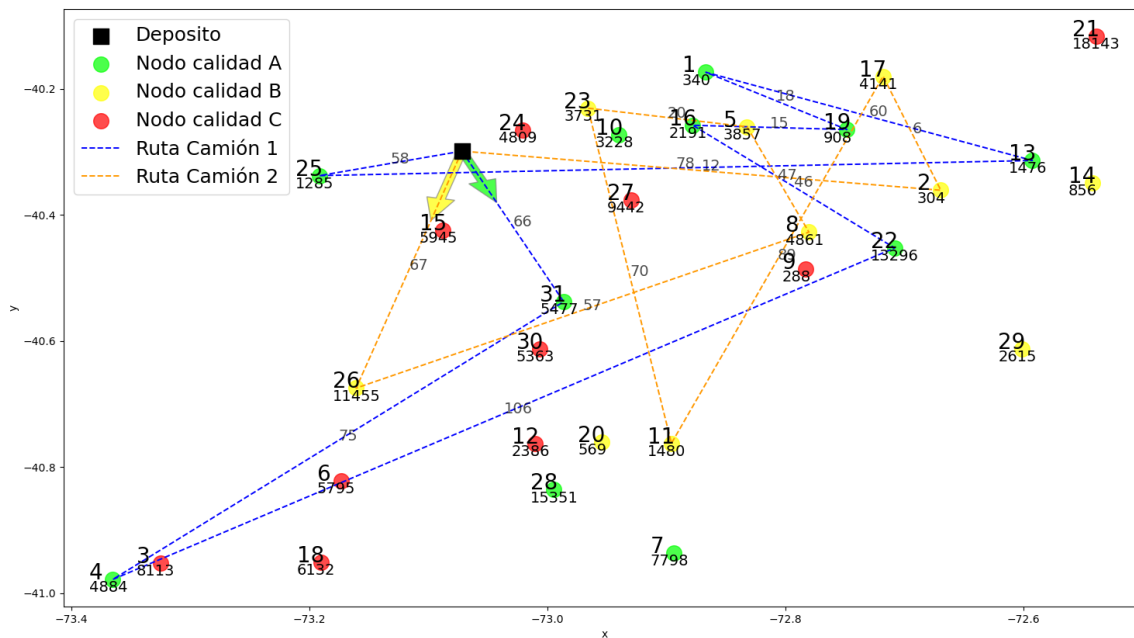


Figura 20: Rutas generadas a partir del algoritmo G-A.  
Fuente: Elaboración propia.

La instancia r4 permite la utilización de tres camiones, pero el resultado del algoritmo de construcción solo utiliza dos camiones. Esto se debe a que cuando termina la construcción de la ruta para el camión 2 se alcanza una solución factible y el algoritmo se detiene. Esta configuración obtiene un valor de 344 en la función objetivo, que se descompone en un beneficio de 1234u por la recolección de leche y 890u por el costo de transporte.

**Etapas de diversificación:** Luego de obtener una solución inicial factible, se realizan perturbaciones iterativamente a la solución actual en la etapa de diversificación, que tiene como objetivo modificar la solución actual y encontrar otra región del espacio de soluciones. La figura 21 muestra el resultado de las modificaciones a las rutas en la etapa de diversificación. La estructura del gráfico es igual a la de la figura 20, pero ahora la etapa de diversificación añade la ruta del tercer camión, que se representa con una línea segmentada de color púrpura. Se observa que el camión 1 inicia su ruta en la planta, para luego visitar en orden a los nodos 2, 13, 16, 1, 25, 10, 20, 29 y 22, recolectando 25304 litros de leche de calidad B. Mientras que el camión 2 visita en orden los nodos 17, 8, 26 y 7, recolectando 28255 litros de calidad B. Finalmente el camión 3 visita en orden los nodos 28, 4 y 31, recolectando 25712 litros de calidad A.

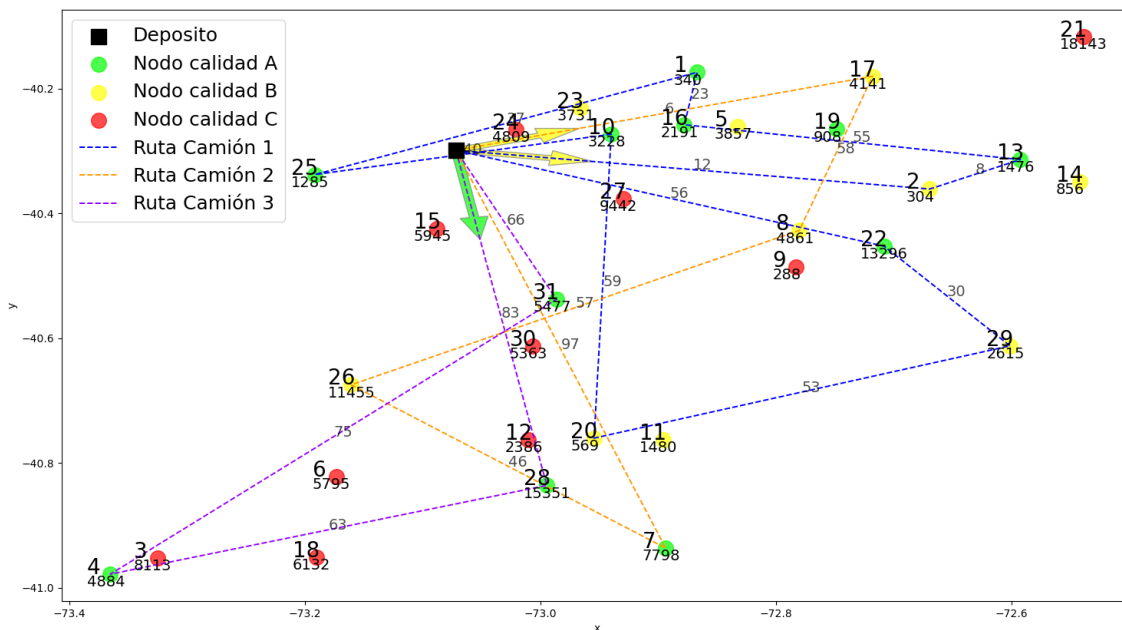


Figura 21: Configuración de rutas luego de la etapa de diversificación.

Fuente: Elaboración propia.

La gran mayoría de los nodos de la solución entregada por la etapa de construcción fueron modificados en la etapa de diversificación, incluso se añade una nueva ruta de recolección para mejorar la calidad de la solución. Esta configuración de rutas obtiene un valor de 868 en la función objetivo, que se descompone en  $1792u$  por beneficio de recolección y  $924$  por el costo de transporte.

**Etapas de intensificación:** Al finalizar la etapa de diversificación, la solución actual es modificada por los operadores de la etapa de intensificación. Estos operadores tienen como objetivo encontrar ligeras modificaciones que permitan mejorar la calidad de la solución, sin alterar el largo de la ruta de cada camión. En la figura 22 se observa como fueron modificadas las rutas desde la etapa de diversificación. En el caso de la ruta del camión 1, el nodo 1 fue reemplazado por el nodo 23, lo que permite aumentar la recolección de leche de calidad A en 3391 litros. Mientras que en la ruta del camión 3 se intercambia el orden de recolección entre los nodos 28 y 4, lo que permite un ahorro de  $1u$  en el costo de transporte sin alterar la cantidad ni la calidad de leche recolectada. Con estas modificaciones, el valor de la función objetivo obtenido por la configuración de rutas en la etapa de intensificación es de 959, que se descompone en un beneficio de  $1863u$  por la recolección y un costo de transporte equivalente a  $905u$ .

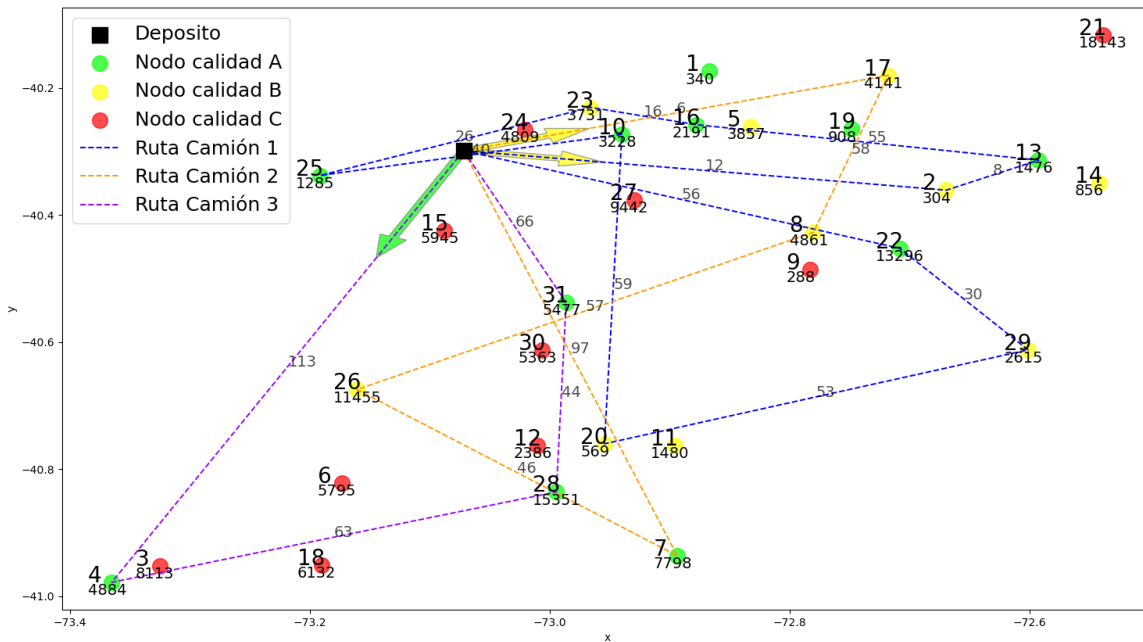


Figura 22: Configuración de rutas luego de la etapa de intensificación.  
Fuente: Elaboración propia.

Se observa que cada etapa aporta diversas modificaciones a las rutas de recolección, que se ven reflejadas en un aumento significativo del beneficio económico obtenido por el proceso de recolección de leche con selección y mezcla.

## 4.9. Conclusiones

En este capítulo se presentan los resultados de los experimentos realizados con los distintos algoritmos diseñados, para encontrar soluciones al Milk Collection Problem with Selection and Blending (MCPSB).

En primer lugar se presentan las instancias utilizadas en los experimentos. Se utiliza un conjunto de instancias pequeñas de pruebas, y un conjunto de instancias reales de hasta 500 nodos productores de leche. Se define el entorno experimental y el método de sintonización de los valores de los parámetros utilizados en las técnicas metaheurísticas.

Luego se presentan los resultados de los experimentos realizados sobre todas las instancias para cada algoritmo diseñado. Las técnicas implementadas fueron diseñadas iterativamente, enfrentando los desafíos y problemas encontrados durante la búsqueda de soluciones para el MCPSB. Se realiza una comparación entre los mejores resultados de las distintas técnicas, en donde se obtienen valores máximos para cada instancia en distintos algoritmos, es decir, ninguna de las técnicas diseñadas supera a las otras en todos los grupos de instancias.

Se experimenta intensivamente con la técnica que obtiene los mejores resultados para las instancias más grandes trabajada. Los resultados obtenidos se comparan con trabajos anteriores realizados sobre el MCPSB, que utilizan las mismas instancias. La técnica es capaz de igualar algunos de los mejores valores encontrados anteriormente. Mientras que para 8 de las 12 instancias reales se encuentran nuevos valores máximos. Finalmente, se destaca que la técnica es sumamente eficiente en términos de recursos computacionales, para la instancia más grande con demanda de leche, que se compone de 500 nodos, se igualan los resultados de otros trabajos en apenas 0.5 segundos de ejecución, en comparación con 6 horas de ejecución. Y para la misma instancia, se encuentran nuevos máximos con mejoras de hasta un 6.6 % en la función objetivo en solo 12 minutos. Mientras que para la instancia de 500 nodos sin restricción de demanda de leche, se encuentran nuevos valores máximos con una diferencia de hasta un 29.7 %.

## CAPÍTULO 5

### CONCLUSIONES

En esta memoria, se aborda el diseño e implementación de varias técnicas metaheurísticas, para encontrar soluciones al problema de optimización Milk Collection Problem with Selection and Blending (MCPSB). Este problema consiste en encontrar un conjunto de rutas eficientes de una flota de camiones, para la recolección de leche de distintas calidades, que son requeridas en la operación de una planta procesadora de productos lácteos. La calidad de las soluciones encontradas se miden con una función de evaluación, que considera minimizar el costo de la recolección por parte de los camiones, y maximizar el beneficio obtenido por la leche recolectada de cada calidad. Se permite la mezcla de leche de distintas calidades dentro de los camiones y en la planta, esto añade un punto de complejidad al problema, al manejar el resultado de las distintas calidades de leche mezcladas.

En los primeros capítulos se define el problema matemático que se aborda, sus restricciones y variables. Luego se presenta una revisión de las técnicas existentes en el estado del arte, en donde se muestran soluciones implementadas anteriormente para el MCPSB. Al ser una problema relativamente nuevo, ha sido abordado en solo tres ocasiones, por lo que también se estudia el Vehicle Routing Problem, un problema similar de enrutamiento de vehículos, del cual se origina el MCPSB. De esta forma, se estudian cuales fueron los problemas y desafíos que tuvieron los investigadores al diseñar algoritmos y técnicas que permiten resolver este problema de optimización.

En los capítulos 4 y 5, se diseña e implementan distintas versiones de una técnica metaheurística para la búsqueda de soluciones para el MCPSB. Comenzando con un algoritmo Greedy que permite construir soluciones iniciales factibles, esta implementación otorga malos resultados en términos de la función objetivo, pero las soluciones se generan rápidamente y son utilizadas en los procesos posteriores. Una vez obtenida la solución inicial, se utiliza una implementación basada en Simulated Annealing (SA) que permite realizar búsqueda local, aplicando movimientos a la solución actual. Se implementan movimientos que añaden y quitan nodos (granjas) de las rutas de los camiones, con el objetivo de modificar las rutas y encontrar otras más eficientes en términos de costos y beneficios. Se analizan los aportes de cada versión de la técnica con resultados y figuras ilustrativas. Finalmente, se implementa un algoritmo de 2 etapas, con una etapa de diversificación basada en SA con los movimientos Add y Drop, y una fase de intensificación basada en Hill Climbing con los movimientos Swap Interno y Swap Externo. Se realizan mejoras al scheduling de temperatura utilizados en SA, y finalmente se mejora el proceso de intercambio de etapas, para enfrentar el problema del estancamiento de soluciones en una región del espacio de soluciones.

En el capítulo 5, se realiza un análisis y comparación de los resultados con trabajos anteriores que abordan el MCPSB. Se igualan algunos valores óptimos conocidos para las instancias pequeñas. Y se encuentran nuevos valores máximos en las instancias reales de mayor can-

tividad de nodos y camiones. Los mejores resultados, y el mayor aporte de esta memoria, es cuando se trabaja con instancias grandes, de hasta 500 nodos. Aquí, se obtienen resultados con valores hasta 30 % mayores en comparación con las mejores soluciones conocidas. Por otro lado, la técnica es sumamente eficiente en términos de tiempo computacional. Para las instancias más grandes, se necesitan solo 0.5 segundos de ejecución para encontrar una solución de buena calidad, en comparación con 6 horas de ejecución requeridas en otras implementaciones. Por último, los mejores resultados fueron encontrados con tiempos de ejecución cercanos a 12 minutos.

Esta implementación puede ser utilizada y mejorada para realizar algoritmos de búsqueda en tiempo real para sistemas de enrutamiento de vehículos, dado que en menos de un segundo es capaz de entregar buenas soluciones para problemas de optimización de rutas con muchos nodos, varios camiones y productos de distinta calidad. Se pueden realizar pruebas y modificaciones al algoritmo, para analizar como se comporta en diversos problemas de enrutamiento con tiempos de cómputo de menos de 1 un segundo.

Finalmente, como trabajo futuro, se propone modificar el algoritmo para mejorar los resultados en instancias pequeñas con poca holgura, en términos del exceso de capacidad de los camiones recolectores. Para esto, se propone mejorar la heurística del algoritmo de construcción de soluciones, como utilizar más información de la instancia y las distancias entre cada nodo al momento de elegir las mejores rutas para cada camión. De esta manera, se podrían realizar menos iteraciones en el proceso de búsqueda local, para dar más recursos a generar soluciones distintas que permitan encontrar mejores valores en otras regiones del espacio de soluciones.

## REFERENCIAS BIBLIOGRÁFICAS

- [FAO, 2016] (2016). Fao.org.
- [Aarts et al., 2003] Aarts, E., Aarts, E. H., y Lenstra, J. K. (2003). *Local search in combinatorial optimization*. Princeton University Press.
- [Abkowitz y Cheng, 1988] Abkowitz, M. y Cheng, P. D.-M. (1988). Developing a risk/cost framework for routing truck movements of hazardous materials. *Accident Analysis & Prevention*, 20(1):39–51.
- [Archetti et al., 2011] Archetti, C., Feillet, D., Gendreau, M., y Speranza, M. G. (2011). Complexity of the vrp and sdvrp. *Transportation Research Part C: Emerging Technologies*, 19(5):741–750.
- [Balas, 1989] Balas, E. (1989). The prize collecting traveling salesman problem. *Networks*, 19(6):621–636.
- [Bodin, 1983] Bodin, L. (1983). Routing and scheduling of vehicles and crews, the state of the art. *Comput. Oper. Res.*, 10(2):63–211.
- [Caramia y Guerriero, 2010] Caramia, M. y Guerriero, F. (2010). A milk collection problem with incompatibility constraints. *Interfaces*, 40(2):130–143.
- [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., 2015] Dayarian, I., Crainic, T. G., Gendreau, M., y Rei, W. (2015). A branch-and-price approach for a multi-period vehicle routing problem. *Computers & Operations Research*, 55:167–184.
- [Dirección General de Transporte Terrestre, 2014] Dirección General de Transporte Terrestre, D. (2014). Observatorio de costes del transporte de mercancías por carretera.
- [Gillett y Miller, 1974] Gillett, B. E. y Miller, L. R. (1974). A heuristic algorithm for the vehicle-dispatch problem. *Operations research*, 22(2):340–349.
- [Glover y Kochenberger, 2006] Glover, F. W. y Kochenberger, G. A. (2006). *Handbook of metaheuristics*, volumen 57. Springer Science & Business Media.
- [Incorporated, 2014] Incorporated, G. (2014). National milk drug residue data base fiscal year 2014 annual report.

- [Montero *et al.*, 2019] Montero, E., Canales, D., Paredes-Belmar, G., y Soto, R. (2019). A prize collecting problem applied to a real milk collection problem in Chile. En *2019 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1415–1422. IEEE.
- [Paredes-Belmar *et al.*, 2017] Paredes-Belmar, G., Lüer-Villagra, A., Marianov, V., Cortés, C. E., y Bronfman, A. (2017). The milk collection problem with blending and collection points. *Computers and electronics in agriculture*, 134:109–123.
- [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.
- [Pires *et al.*, 2011] Pires, A., Martinho, G., y Chang, N.-B. (2011). Solid waste management in European countries: A review of systems analysis techniques. *Journal of environmental management*, 92(4):1033–1050.
- [Reed *et al.*, 2014] Reed, M., Yiannakou, A., y Evering, R. (2014). An ant colony algorithm for the multi-compartment vehicle routing problem. *Applied Soft Computing*, 15:169–176.
- [Riff y Montero, 2013] Riff, M.-C. y Montero, E. (2013). A new algorithm for reducing metaheuristic design effort. En *2013 IEEE Congress on Evolutionary Computation*, pp. 3283–3290. IEEE.
- [Rojas Molina, 2005] Rojas Molina, I. (2005). Análisis y mejora del sistema de recogida de leche de una industria láctea.
- [Sbihi y Eglese, 2007] Sbihi, A. y Eglese, R. W. (2007). Combinatorial optimization and green logistics. *4OR*, 5(2):99–116.
- [Soto, 2019] Soto, C. (2019). Un acercamiento meta-heurístico para el problema de recolección de leche con selección y mezcla. unpublished.
- [Toth y Vigo, 2014] Toth, P. y Vigo, D., editores (2014). *Vehicle Routing: Problems, Methods, and Applications, Second Edition*. Número 18 en MOS-SIAM Series on Optimization. SIAM.
- [Van Laarhoven y Aarts, 1987] Van Laarhoven, P. J. y Aarts, E. H. (1987). Simulated annealing. En *Simulated annealing: Theory and applications*, pp. 7–15. Springer.
- [Wikipedia, ] Wikipedia. Knapsack problem. [https://en.wikipedia.org/wiki/Knapsack\\_problem](https://en.wikipedia.org/wiki/Knapsack_problem). Accessed: 2020-10-17.
- [Wolpert y Macready, 1997] Wolpert, D. H. y Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82.
- [Yu y Yang, 1998] Yu, G. y Yang, J. (1998). Optimization applications in the airline industry. En *Handbook of combinatorial optimization*, pp. 1381–1472. Springer.