2019-08

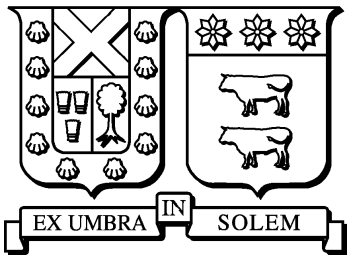
UN ENFOQUE DE ALGORITMO GENÉTICO APLICADO A UN PROBLEMA DE RECOLECCIÓN DE LECHE EN CHILE

ESCOBAR BOEHMWALD, DIEGO

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE INFORMÁTICA

SANTIAGO – CHILE



"UN ENFOQUE DE ALGORITMO GENÉTICO APLICADO A UN PROBLEMA DE RECOLECCIÓN DE LECHE EN CHILE"

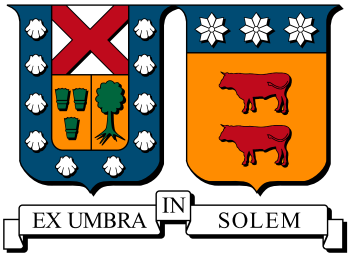
DIEGO ESCOBAR BOEHMWALD

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INFORMÁTICO

PROFESOR GUÍA: ELIZABETH MONTERO URETA

AGOSTO 2019

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO – CHILE



"UN ENFOQUE DE ALGORITMO GENÉTICO APLICADO A UN PROBLEMA DE RECOLECCIÓN DE LECHE EN CHILE"

DIEGO ESCOBAR BOEHMWALD

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL INFORMÁTICO

PROFESOR GUÍA: ELIZABETH MONTERO URETA
PROFESOR CORREFERENTE: JOSE LUIS MARTÍ LARA

AGOSTO 2019

Resumen

En este artículo se trata un problema real de recolección de leche en el sur de Chile, donde se trabaja con una empresa procesadora de leche la cual requiere recolectar leche parar poder elaborar sus productos, dicha empresa posee una flota heterogenia de camiones destinados para la recaudación de la leche. La cantidad total de leche diaria a recolectar es designada por la empresa procesadora, recolectar una mayor cantidad a la designada genera un costo sobre demanda.

El objetivo de este trabajo es encontrar una solución óptima, es decir, tratar de encontrar las rutas más óptimas para que los camiones realicen la recolección, minimizando el costo total reduciendo la cantidad de kilómetros recorridos por los vehículos y recolectando la menor cantidad posible de leche sobre el mínimo diario. Con este propósito se busca resolver el problema a través de un algoritmo genético, dicho algoritmo fue implementado con distintas herramientas (distintos armados iniciales, búsqueda local y distintos métodos de selección en el cruzamiento), las cuales fueron probadas en distintas configuraciones buscando cual resulta más efectiva para resolver este problema. Todos los experimentos fueron probados en casos del mundo real. Se compararon los rendimientos de todas las configuraciones probadas, en donde se obtuvieron soluciones de buena calidad en la gran mayoría de los casos, para instancias pequeñas el algoritmo converge rápidamente a una única solución, mientras que en instancias grandes las configuraciones mostraron un margen mayor de soluciones, es en estos casos que se puede requerir otros enfoques o rediseños para obtener mejores resultados.

Abstract

This article deals with a A prize collecting problem applied to a real milk collection problem in Chile, we work with a milk processing company which requires milk to be collected in order to be able to produce its products, said company has a heterogeneous fleet of trucks destined for collection of milk. The total amount of daily milk to be collected is designated by the processing company, collecting a larger amount of the designated couta generates a cost on demand.

The goal of this work is to find an optimal solution, that is, to try to find the most optimal routes for the trucks to carry out the collection, minimizing the total cost by reducing the amount of kilometers traveled by the vehicles and collecting as little milk as possible over the daily minimum. For this purpose, the problem is sought through a genetic algorithm, said algorithm was implemented with different tools (different initial assemblies, local search and different selection methods at the crossing), which were tested in different configurations looking for which is more efficient. All experiments were tested in real world cases. The yields of all the tested configurations were compared, where good quality solutions were obtained in the vast majority of cases, for small instances the algorithm quickly converges to a single solution, while in large instances the configurations showed a greater margin of solutions, it is in these cases that other approaches or redesigns may be required to obtain better results.

Índice de Contenidos

Índice de cuadros

Índice de figuras

Glosario

- Algoritmo: Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema.
- Cruzamiento: Combinación de dos soluciones para encontrar una nueva solución.
- Elitismo: Reserva y re-introduce las mejores soluciones en la población.
- Función de evaluación: Medida cuantitativa que representa una o más funciones medidas cuantitativa del funcionamiento del sistema que se desea optimizar,o bien la cantidad de restricciones satisfechas.
- Instancia: Una versión del problema con ciertas características definidas.
- Configuración: Una forma de resolver el problema con ciertas características definidas.
- Grafo: Un conjunto de nodos unidos por arcos que representan relaciones entre los nodos.
- Mutación: Transformación aleatoria de una solución.
- Restricción: Relación entre las variables que restringen los valores que éstas pueden tomar.
- Solución: Conjunto de valores para las variables que satisfacen las restricciones de un problema.
- Variable: Decisiones que se pueden tomar para afectar el valor de la función objetivo.

- Exploración: Buscar soluciones enfocado en amplitud, es decir, concentrarse en encontrar diferentes soluciones en vez de buscar buenas soluciones.
- Explotación: Buscar soluciones enfocado en profundidad, es decir, concentrarse en encontrar mejores soluciones en vez de buscar soluciones diferentes.

Introducción

La leche es uno de los alimentos más consumidos por el hombre, perteneciendo a su canasta de comida diaria, por lo que la recolección de este alimento desde cada granja productora es importante como un paso intermediario desde su producción hasta su consumo. Dicha recolección consta de un transporte de la leche el cual tiene un proceso logístico que debe ser cumplido con eficiencia, si este no fuese el caso podría implicar costos elevados para las empresas que recolectan la leche, a tal punto que solo el coste de transporte puede llegar a representar alrededor del 30 % del costo final de la leche, por lo que es de gran relevancia encontrar formas de mejorar este proceso de recolección y transporte.

El proceso de recolección y transporte sera basado en un grupo de camiones que ira recolectando la leche de granja en granja hasta cumplir una cuota diaria designada por la planta procesadora, modelo el cual tiene un gran parecido a *Prize Collecing Vehicle Routing Problem* (*PCVRP*). Este problema fue evaluado y será afrontado mediante un algoritmo genético (*AG*), este tipo de algoritmo imita el proceso de evolución de un grupo de seres vivos, mejorando en cada generación donde cada hijo obtiene las mejores características de sus padres.

Capítulo 1

Definición del Problema

En este trabajo se estudia el problema de la recolección de leche a lo largo del sur de Chile. Las granjas productoras de leche están localizadas en sectores rurales que tienen accesos únicos. Dichas granjas pueden vender su producto a distintas distribuidoras o bien estar asociadas a una cooperativa lechera. Algunas plantas procesadoras pueden poseer pequeñas flotas heterogéneas de camiones que compren y transporten ciertas cantidades de leche en busca de minimizar costos de transporte. Este trabajo de memoria se enfocará en este modelo de negocios. Se establece que los camiones compren el total del producto a las granjas visitadas debido a políticas implementadas para proteger a las granjas productoras y siempre se debe comprar el producto de las granjas visitadas. Además se considera que la leche recolectada en todas las granjas es de calidad homogénea. La planta procesadora impone una cuota diaria de leche recolectada que debe ser cumplida, cualquier monto de leche recolectada sobre esta cuota se considera una perdida. Esto, debido a que la leche recolectada implica un costo de compra para la planta procesadora y, al sobrepasar la cuota, significa que la planta no la necesita. Todos los camiones deben comenzar y terminar su recorrido diario en la planta procesadora.

El problema de recolección de leche viene estudiándose desde hace algunos años. Dentro de las principales versiones estudiadas se consideran variaciones como:

- Distintos niveles de calidades de leche.

- Tiempo antes de que el producto se estropee.
- Ubicación de puntos de recolección.
- Ventanas de tiempo para la recolección.
- Recolecciones periódicas.

La variante relacionada a las distintas calidades de leche considera que cada granja produce leche de un nivel de calidad específico. Distintos niveles de calidades de leche son requeridos para producir productos específicos en la planta procesadora. Así, distintas cuotas son requeridas para cada tipo de leche definido en el problema. Dos opciones se estudian en este contexto, una variante que no permite mezclas y una que sí las permite. En caso de no considerar mezclas es posible considerar camiones con compartimientos, los cuales recolectan leche de distintas calidades y llevan este producto de manera separado en los compartimientos del camión. En este caso cada camión posee una capacidad máxima para cada tipo de leche transportada. En caso de considerar mezclas, distintos tipos de leche pueden ser cargados en un camión, pero el resultado de esta mezcla produce un deterioro en el total de leche transportada en el camión.

La segunda variante considera los tiempos entre la recolección y entrega de la leche en la planta procesadora. Esto considerando que los camiones no poseen sistemas de refrigeración adecuados para realizar recorridos muy largos. El tiempo necesario para que el producto se estropee varia de problema a problema. Una variante de este problema considera además que los camiones pueden realizar escalas a sitios con sistemas de enfriamiento, lugares donde la leche será enfriada, manteniendo la cadena de frío. Así, es posible extender los tiempos de deterioro del producto a costa de recorrer una mayor distancia a los puntos de enfriamiento.

La tercera variante considera que en algunos problemas los productores de leche pueden llevar su producto a distintos punto de recolección, este transporte utiliza vehículos de menor capacidad reduciendo los costos de transporte y facilitando la recolección de distintas granjas productoras. Este procedimiento generalmente se utiliza para favorecer a los productores de leche que producen menores cantidades del producto y, por tanto, no son visitados.

La cuarta variante considera que algunas granjas productoras tienen ventanas de tiempo específicas disponibles para la recolección de su producto, por lo que, al momento de calcular las rutas de recolección, se debe tener en consideración el instante de tiempo en que se consigue visitar las distintas granjas para recoger el producto en un horario donde estas estén disponibles.

Para la ultima variante se considera que el planeamiento de la recolección de leche no se realiza de manera diaria si no en un intervalo de tiempo más amplio, usualmente semanas. En general se considera que la producción de las granjas tiende a ser más compleja, en sus intervalos de disponibilidad horaria y nivel de producción lista para retirar, por lo que es común ver esta ultima variante junto a alguna de las anteriormente mencionadas.

En este documento se tratara una recolección de leche estándar, es decir, la recolección del producto es diaria, no se considera el tiempo de deterioro de la leche, la calidad del producto es homogéneo, las granjas están siempre disponibles para recolectar el producto. Además no se consideran puntos de recolección, por lo que hay que visitar cada granja si se requiere obtener el producto de dicha productora.

La figura 1.1 muestra un esquema de una posible solución del problema acá estudiado. En este ejemplo, la planta P (diamante central) requiere una cuota de 330 litros de leche. Las seis granjas están representados con círculos, numeradas del 1 al 6, y sus producciones son los números que se muestran arriba de cada círculo. En este caso particular se considera la disponibilidad de dos camiones. Usando dos camiones con capacidad de 150 y 220 litros, se obtienen dos rutas. El primer vehículo, cuya ruta se ha dibujados con líneas rojas, recoge la leche de las granjas 1 y 4. Comienza su recorrido en la planta procesadora, luego visita la granja 1, a continuación visita la granja 4 y finalmente vuelve a la planta procesadora. Al visitar una granja se debe recoger la producción completa de dicha granja. Así, el camión 1, recolecta una cantidad total de 150 litros de leche ocupando toda su capacidad.

El segundo camión, cuya ruta fue dibujada de color púrpura, comienza su recorrido en la planta procesadora, recoge la leche de las granjas 2, 3 y 6 en ese orden y luego vuelve a la planta procesadora. El total de leche recolectada es de 200 litros de leche ocupando casi toda su capacidad.

En la solución propuesta la granja 5 no es visitada. En este ejemplo la elección de granjas es basada en aquellas más cercanas y que, en conjunto provean la mejor combinación que satisface la cuota de leche requerida. Cabe notar que el resultado de este ejemplo es una solución factible. Esto debido a que, primero, la cantidad de leche recolectada por los dos camiones satisface la cuota solicitada por la planta procesadora, y, segundo, no se sobrepasa la capacidad de transporte de cada uno de los camiones.

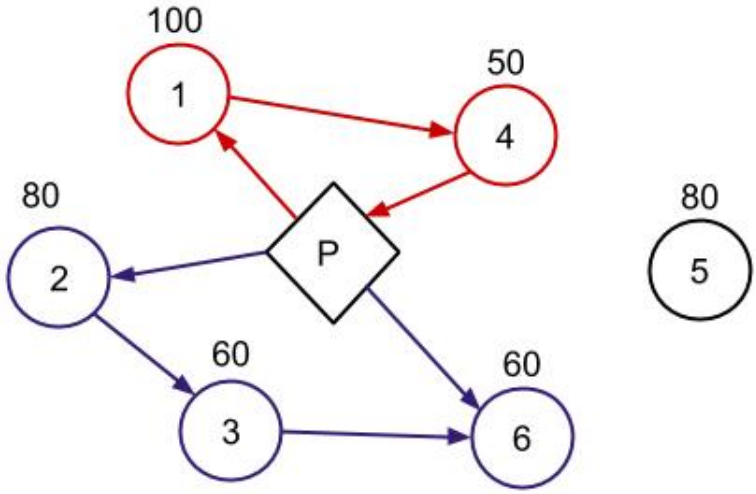


Figura 1.1: Ejemplo de recolección de leche con dos camiones.

1.1. Modelo matemático

A continuación se presenta el modelo matemático del problema a resolver. Este modelo fue extraído del artículo de Montero *et al.* [9]. Primero se listan los parámetros del problema, seguido de las variables función objetivo y las restricciones explicadas en detalle.

1.1.1. Parámetros

La tabla 1.1 resume los parámetros del modelo matemático propuesto.

El modelo considera la representación del problema como un grafo completo que considera como nodos la planta procesadora y el conjunto de granjas. Se considera como nodo 0 la

Parámetro	Descripción
$G(N_0, A)$	Grafo completo.
N_0	Conjunto de nodos. Considera granjas y planta procesadora.
A	Conjunto de arcos.
N	Conjunto de nodos. Considera sólo granjas.
d_{ij}	Matriz de distancias entre nodos. Considera granjas y planta procesadora.
C	Costo de transporte por kilómetro.
L_i	Costo por litro de leche de la granja i.
q_i	Producción granja i.
P	Producción requerida por la planta.
K	Número de camiones.
Q_k	Capacidad de los camiones. $k \in K$

Cuadro 1.1: Parámetros modelo matemático.

planta procesadora. El conjunto de arcos A considera todas las distancias entre todas las granjas del problema y planta procesadora. Se considera $A = \{i \in N_0, j \in N_0 : i \neq j\}$. N considera solo el conjunto de granjas.

Por otro lado se consideran los costos de transporte. Para esto se considera la matriz de distancias d_{ij} que corresponde a la distancia (en kilómetros) sobre los arcos $(i, j) \in A$. C corresponde al costo de transporte por kilómetro.

Respecto a los costos de recolección se considera el costo por litro de leche L_i en la granja $i \in N$. La producción de cada granja viene dada por q_i, para cada granja $i \in N$. P corresponde a la cuota de leche requerida por la planta procesadora. Q_k corresponde a la capacidad de cada camión $k \in K$.

1.1.2. Variables

El modelo matemático requiere la definición de tres conjuntos de variables. La primera, en la ecuación 1.1, variable binaria relacionada con las visitas de los camiones a los nodos.

La segunda, en la ecuación 1.2 variable binaria relacionada con la carga de leche de los vehículos en las visitas. La última variable, en la ecuación 1.3, es una variable real que permite controlar el nivel de leche recolectada y la aparición de sub-ciclos en las rutas de recolección.

$$x_{ij}^k = \begin{cases} 1 & \text{si el camión } k \text{ viaja desde un nodo } i \text{ a un nodo } j \\ 0 & \text{si no} \end{cases} \tag{1.1}$$

$$y_i^k = \begin{cases} 1 & \text{si el camión } k \text{ carga leche de la granja } i \\ 0 & \text{si no} \end{cases} \tag{1.2}$$

$$y_i^k = \text{carga de leche obtenida desde la granja } j \text{ utilizando el vehículo } k \tag{1.3}$$

1.1.3. Función objetivo

La función objetivo minimiza tanto el costo de transporte como el de la demanda de leche. La primera parte de la ecuación calcula la distancia total recorrida por todos los camiones. La segunda parte suma el costo total asociado a la leche recolectada.

$$\text{Mín} \quad C \sum_{k \in K} \sum_{(i,j) \in A} d_{ij} x_{ij}^k + \sum_{k \in K} \sum_{i \in N} L_i q_i y_i^k \tag{1.4}$$

1.1.4. Restricciones

A continuación se listan y describen las restricciones que considera el problema de recolección de leche estudiado.

La ecuación (1.5) establece que no todas las granjas necesitan ser visitadas. Además establece que cada granja puede ser visitada por a lo más un camión.

$$\sum_{k \in K} y_i^k \leq 1 \quad \forall i \in N \tag{1.5}$$

Además, si la decisión es de recolectar leche de una granja, entonces toda la producción es cargada, osea, no se puede recolectar en fracciones. Esta restricción es establecida para proteger a los productores asegurando que vendan toda su producción.

Las restricciones en la ecuación (1.6) controlan la compatibilidad entre las cargas desde las granjas y las rutas de los vehículos.

$$\sum_{h \in N_0:(h,i) \in A} x_{hi}^k = y_i^k \quad \forall i \in N, k \in K \tag{1.6}$$

Las restricciones en la ecuación (1.7) aseguran la utilización total de la flota de camiones. Esta restricción puede ser relajada cuando la capacidad total de los camiones es mucho mayor a los requerimientos de leche de la planta procesadora.

$$\sum_{j \in N:(0,j) \in A} x_{0j}^k = 1 \quad \forall k \in K \tag{1.7}$$

Las restricciones en la ecuación (1.8) controlan el balance de flujo de cada nodo y camión.

$$\sum_{i \in N_0:(i,j) \in A} x_{ij}^k = \sum_{h \in N_0:(j,h) \in A} x_{jh}^k \quad \forall k \in K, j \in N_0 \tag{1.8}$$

Las restricciones en la ecuación (1.9) controlan la satisfacción de la recolección mínima diaria de la planta.

$$\sum_{i \in N} \sum_{k \in K} q_i y_i^k \geq P \tag{1.9}$$

Las restricciones en la ecuación (1.10) controlan que se respete la capacidad de recolección de los camiones.

$$\sum_{i \in N} q_i y_i^k \leq Q^k \quad \forall k \in K \tag{1.10}$$

Las restricciones en la ecuación (1.11), conocidas como las restricciones de *Miller-Tucker-Zemlin* controlan la aparición de sub-ciclos en las rutas de los camiones.

La cantidad de leche en cada paso es usado para secuenciar apropiadamente las rutas.

$$\begin{aligned} T_j^k &\geq T_i^k + q_i - Q^k(1 - x_{ij}^k) \\ &\forall k \in K, i \in N, j \in N_0 : (i,j) \in A; \end{aligned} \tag{1.11}$$

Finalmente, las restricciones en las ecuaciones (1.12) y (1.13) controlan el dominio de las variables de decisión.

$$x_{ij}^{k}, y_{i}^{k} \in \{0, 1\} \quad \forall (i, j) \in A, k \in K \tag{1.12}$$

$$T_{j}^{k} \geq 0 \quad \forall j \in N_0, k \in K \tag{1.13}$$

1.2. Resumen

El trabajo estudia un problema detectado en la cadena de recolección de leche en el sur de Chile. Hay que considerar que una planta procesadora de leche necesita recolectar diariamente una cierta cantidad de este producto. Para esto se provee de una flota heterogénea de camiones que visitan distintas granjas productoras en las que compran y recolectan la totalidad de leche disponible. La calidad de este producto es homogénea, independiente de la granja productora, y cualquier cantidad de leche recolectada que exceda la demanda diaria, se transforma en pérdida.

A partir de lo anterior, este trabajo propone un algoritmo que obtenga soluciones las cuales busquen minimizar el costo de la planta procesadora de leche. Esto es, el armado de soluciones que considere, por un lado, rutas de recolección más eficientes que disminuyan la distancia recorrida, y por otro lado, el ajuste de las cantidades de leche recolectada para satisfacer la demanda, disminuyendo, con ello, las pérdidas del producto.

Capítulo 2

Estado del Arte

El problema de recolección de leche es un problema de ruteo de vehículos que incorpora, además de los costos de desplazamiento, ganancias por las visitas a las granjas. Estas ganancias se entienden como la cuota mínima a satisfacer por el proceso de recolección. Una vez atendida la cuota mínima, la leche adicional genera un costo, el cual es perjudicial para la calidad de la solución. El problema clásico de ruteo de vehículos se conoce como VRP (Vehicle Routing Problem). El VRP consiste en la distribución de un cierto bien desde un punto conocido como depósito a un conjunto de nodos clientes. El VRP es una combinación de varios problemas tipo vendedor viajero, por ello, su complejidad NP-duro ha sido demostrada en base al TSP. Existen muchas variantes para el VRP, algunas consideran múltiples depósitos, ventanas de tiempo, información dinámica, entre otros. El PCVRP es un tipo de problema VRP que considera premios asociados a las visitas. Esto se asemeja mucho a la situación que se tiene en el problema de recolección de leche en que se obtiene una ganancia asociada a las visitas a granjas. Por último, existen algunas aproximaciones específicas a problema de recolección de leche que consideran leches de un tipo y diferentes tipos, uso de contenedores, entre otros.

Así, este estado del arte presenta primero un resumen de los principales acercamientos para el VRP, luego algunos acercamientos interesantes relacionados al PCVRP para finalmente ahondar en acercamientos específicos relacionados con el problema de recolección de leche.

El VRP es uno de los problemas más antiguos a resolver, fue introducido por primera vez en 1959 por Dantzig y Ramser como "Problema de Despacho de Camiones" (Truck Dispatching Problem) en donde se modelaba una flota de camiones homogénea que podría asistir en el transporte de petróleo en las bencineras. De este problema nació la primera versión del VRP, el cual consiste en satisfacer a los clientes, los cuales estaban geográficamente dispersos alrededor de la fábrica central con una flota de camiones heterogénea. VRP es uno de los problemas más populares existentes, debido a esto se han implementado muchas formas de resolverlo, y además, se han estudiado muchos casos particulares con restricciones asociadas a la vida real tales como:

- Tiempos de viaje dependientes del tiempo, considerando congestión del tráfico.
- Intervalos limitados de tiempo para recolectar y entregar el producto.
- Información dinámica, que cambia a través del tiempo, sobre las condiciones del problema.

VRP ya es por sí solo un problema NP-duro, por lo que revisar casos particulares como lo mencionados anteriormente, entre otros, solo aumenta su complejidad. Debido a esto los autores de distintos trabajos enfocados en casos particulares trabajaron nuevos algoritmos para resolver casos particulares. En general, cada versión del problema posee características particulares que hacen más compleja su resolución, por lo que es difícil encontrar una única propuesta de solución para todo el espectro de problemas VRP[1].

Otro punto importante a mencionar es que, de acuerdo a Eksioglu (2009), la literatura del problema VRP ha ido en un aumento exponencial del 6 % anual, sumando una gran cantidad de aproximaciones al problema. Durante los últimos años se han introducido diferentes variantes al problema con distintos términos, algunos de estos problemas fueron enfrentados de distintas maneras, Wassan y Nagy [13] definieron el problema Vehicle Routing Problem with Pickup and Delivery (VRPPD) mediante un modelo de programación lineal que considera todos los casos del problema. Cordeau *et al.* enfrentan el problema Vehicle Routing Problem with Time Windows (VRPTW) [6] a través de un algoritmo basado en Tabu Search, una búsqueda local metaheurística que explora el espacio de solución moviéndose en cada

iteración desde la solución actual hasta la mejor solución en su vecindario N (s). Cheikh *et al.* [4] proponen un algoritmo basado de búsqueda variable del vecindario, en el que cuatro estructuras de vecindario están diseñadas para encontrar la planificación de recorridos que resuelve el problema Vehicle Routing Problem with Multiple Trips (VRPMT). Siendo VRP consistente (Consistent Vehicule Routing Problem), o bien $ConVRP$, la última iteración de estas variantes mencionadas e implica diseñar rutas de costo mínimo para dar servicio a un conjunto de clientes frecuentes y no frecuentes con demandas conocidas durante varios días a través de un flota homogénea de vehículos capacitados que regresan al depósito, mientras que los vehículos satisfacen la capacidad, duración de la ruta y restricciones de consistencia.

PCTSP es una generalización del problema TSP, donde el vendedor recibe un ”premio”por cada ciudad visitada y recibe una penalidad por cada una de ellas no visitada. Además hay un costo asociado al viaje de cada ciudad, por lo que el objetivo es minimizar el costo de los viajes y las penalidades, pero asegurando la obtención de una cantidad mínima de premios.

En [7] Haouari planteó que para resolver el PCTSP se necesita que el algoritmo genético incorpore el algoritmo de volumen. El algoritmo de volumen es un algoritmo basado en multiplicadores lagrangianos para resolver el problema dual lagrangiano, utilizando el algoritmo de Prim modificado (MPP). El algoritmo de Prim construye al árbol de cobertura de costo mínimo (Minimum spanning tree MST). El algoritmo está basado en tomar un nodo raíz y empezar a agregar nodos iterativamente a la solución mediante algún criterio para minimizar el costo del árbol armado. En este artículo en particular consideran un coeficiente entre el costo de un arco a unir y el valor del nodo seleccionado, por lo que todos los nodos son evaluados en un primer paso, y luego son elegidos en un segundo paso. Esto se realiza hasta que se la suma de todos los nodos evaluados sea mayor o igual a la cuota buscada. Una vez armado el grafo se busca el MST mediante el algoritmo de Prim. Posteriormente se poda el árbol poniendo en un conjunto todas las hojas que, al ser retiradas, permiten que la suma de los premios restantes sea mayor a la cuota buscada. Las hojas se retiran de una en una quitando primero los arcos más pesados sin bajar de la cuota requerida. Las soluciones de este acercamiento son representadas como cromosomas, es decir cadenas binarias del largo de la cantidad de nodos menos uno ($|V| - 1$) indicando que nodos han sido visitados. La población inicial para este algoritmo genético híbrido lagrangiano se construye creando un

50 % de los individuos usando heurísticas lagrangianas y el 50 % restante de manera aleatoria. Luego cada candidato es evaluado utilizando el algoritmo de Prim modificado. Se crean nuevos candidatos usando seleccionados a partir de un torneo binario a los que se les aplica cruzamiento en dos puntos con cierta probabilidad. En la propuesta se evalúan 45 instancias del problema que consideran entre 100 y 500 nodos. Los resultados indican que la propuesta es rápida, robusta y entrega buenos resultados.

En [12] Rojas y Meza proponen un Algoritmo Genético para Prize Collecting Traveling Salesman Problem ($PCSTP$). El algoritmo comienza creando una población inicial de manera pseudo-aleatoria. Las soluciones a través de una representación binaria del tamaño de los vértices posibles, donde cada valor igual a 1 representa que dicho vértice está presente en la solución. Para esto toma un conjunto de nodos con distintos pesos para ingresarlos a la primera solución. Cada nodo ingresado es unido a un nodo ya existente en la solución y el arco es creado. Luego realiza un podado en el árbol cortando cualquier nodo que su valor sea menor que el coste para llegar a él. Es decir un nodo con aporte negativo a la solución. El algoritmo trabaja realizando en cada iteración Selección aleatoria, cruzamiento en dos puntos y mutación Bit flip. El resultado del operador de crossover es descartado si existe una solución similar en la población. La mutación es solo aplicada a los hijos. La propuesta considera un algoritmo genético paralelo, el cual implica en dividir la población en distintas sub-poblaciones o "islas". En un algoritmo genético basado en islas se realizan migraciones esporádicas orientadas a diversificar la población. En este caso, solo los mejores individuos de cada isla se eligen para cada migración. No se permiten soluciones repetidas en una misma isla. La migración de individuos es asincrónica, es decir, ocurre en cualquier momento, y cada isla está conectada de manera unidireccional en topología de anillo. En la propuesta se evalúan instancias del problema que consideran entre 500 y 2500 nodos. Este trabajo concluye que a medida que la cantidad de poblaciones trabajadas es mayor, los resultados mejoran, pero existe un límite para la mejora a través de este método.

En [11] el autor estudió una versión asimétrico del PCSTP mediante un algoritmo Tabu Search. La fase de construcción fue afrontada basada en el procedimiento $GENIUS$ y una búsqueda local 2-opt. El procedimiento $GENIUS$ consiste en dos fases. En la primera fase

denominada GENI (GEnerate an iNItial solution), se crea una solución inicial. En la segunda fase, denominada US, se refina la solución. El procedimiento $GENI$ construye la solución interactivamente agregando nodos al tour. La inserción de un nuevo nodo no necesariamente tiene lugar entre dos nodos adyacentes, más bien considera los nodos que ya están en el recorrido y verifica las rutas posibles que conectan estos nodos. Luego calcula el costo de inserción del nuevo nodo analizando qué arcos se eliminarán y qué segmentos de la ruta serán revertidos. Por lo general, dependiendo de los dos nodos seleccionados, hay dos formas posibles de inserción. La verificación se lleva a cabo para todos los pares de nodos que ya están en el recorrido y se realiza la inserción de menor costo. Luego de que una solución factible es obtenida por la primera fase, se inicia la fase US. En esta fase un nodo es removido de la solución, desligándolo del recorrido. Para el refinamiento de la solución se aplicó el algoritmo Tabu Search, el cual considera tres pasos. En el primer paso se remueve un nodo. En el segundo paso se realiza la agregación de un nodo utilizando $GENIUS$ e intercambio de nodos en la solución. En el tercer paso se realiza un intercambio de nodos, cambiando un nodo que está en los recorridos, por uno que no lo esta. Notar que al inicio de TS se realiza la operación 3-opt, y luego de cada movimiento se realiza una operación 2-opt. En la propuesta se evalúan instancias del problema que consideran entre 31 y 501 nodos. Los resultados indican que la propuesta es favorable en instancias de mayor tamaño que otras aproximaciones como Cluster Search.

A continuación se describen los principales acercamientos del estado del arte relacionados a problemas de recolección de leche. Dentro de estos se destacan autores como Caria [3], quien enfrentó un problema VRP de recolección de leche a través de una herramienta creada con el algoritmo Ant-colony. La herramienta desarrollada en este estudio utilizó la asignación de mapas por GPS y los volúmenes de leche en cada nodo para calcular el costo por litro de leche. El algoritmo basado en colonias de hormigas trabaja liberando muchas hormigas, que inician recorrido desde la fábrica, viajando distancias aleatorias hasta encontrar su objetivo, es decir, las productoras de leche. Las hormigas que se demoren una mayor cantidad de tiempo en regresar a la fábrica, es decir, aquellas que tomaron un recorrido más largo, dejan un rastro de feromonas menor. El algoritmo se probó con información de cinco fabricas de queso ubicadas en Sardinia (Italia). Las pruebas realizadas consideran conjuntos desde 5 a

27 puntos de recolección. Los resultados mostraron que esta herramienta mejoró la eficiencia de la recolección de leche, reduciendo el número de rutas y las distancias de los recorridos.

Otro autor que se enfrenta al problema de recolección de leche es Claassen [5], quien estudió el problema basado en un enfoque de Ruteo de Vehículos Periódico (Periodic Vehicle Routing Problem o PVRP). La principal diferencia entre un PVRP y un VRP tradicional resulta en que el planeamiento del VRP considera sólo un día de recolección mientras que el PVRP consta de T días (siendo T una cantidad arbitraria definida en el problema). Además, en el planteamiento de Claassen, cada granja puede ser visitada más de una vez y por distintos camiones recolectores. El problema de recolección de leche estudiado considera condiciones adicionales a las del PVRP. Así, los autores proponen estudiar una versión del PVRP modificado a las restricciones y condiciones implementadas. Así, resuelven el problema utilizando un modelo de programación entera mixta SOS1 (Special Order Sets type 1) para la asignación de visitas permitidas y estables. En el acercamiento propuesto es capaz se definió que las granjas individuales deberían agruparse en racimos (clusters) y definir las visitas a estos racimos, de encontrar solución a instancias a lo más diez puntos de recolección o racimos, ya que el tiempo de resolución aumentaba exponencialmente luego de considerar el séptimo racimo.

A la fecha no solo se ha intentado resolver problemas de recolección de leche bajo el enfoque del VRP, también se ha enfrentado con distintos enfoques. Durante el 2018 en [10] se trató una variante de problema, Milk collection problem with Blending and Collection Points (MBCP), el cual se basa en que la recolección de varias granjas productoras se realiza en ciertos puntos de recolección, se considera también los puntos de entrega de la leche y que la calidad de los distintos tipos de leche es distinta, una condición que se tomó en este problema es que la leche de distintas calidades puede mezclarse, el resultado de la mezcla se considera leche de la peor calidad agregada en dicha mezcla. En este problema se utilizó un modelo de programación lineal entera mixta para resolver instancias pequeñas usando CPLEX. Se consideran instancias pequeñas hasta 40 granjas o puntos de recolección aproximadamente. Así, se deciden las granjas y puntos de recolección a visitar, la secuencia de las visitas y la cantidad de leche entregada en los puntos de recolección sin que los camiones faciliten

esta entrega. También se propuso una solución de tres fases para instancias grandes la cual consta de, primero, resolver de manera óptima un problema de cobertura para asignar a los pequeños productores a los puntos de recolección, segundo, generar rutas viables utilizando ACO, y por último elegir la mejor ruta generada por la herramienta metaheurística para visitar granjas y puntos de recolección.

Caramia [2] estudió el problema de recolección de leche con restricciones incompatibles, enfocándose en problemas como la inaccesibilidad de camiones grandes en pequeñas granjas. Además considera que la calidad de la leche es diferente en cada granja por lo que se utilizó una flota de camiones con compartimentos para su recolección. Los autores proponen una búsqueda local heurística de dos fases, la primera fase consta de calcular la solución óptima del problema de asignación de ruta del agricultor, obteniendo el subconjunto de agricultores para ser atendido por cada camión, respetando la demanda y limitaciones de capacidad. Sin embargo, porque esta tarea podría no ser factible con respecto a la máxima duración de un recorrido, la segunda fase del algoritmo verifica la duración del recorrido resolviendo la formulación del problema de definición de rutas que minimiza las distancias.

Por último, Montero *et al.* trabajaron el problema Prize collecting problem aplicado a un problema real de recolección de leche en [9]. En su estudio, los autores presentaron un modelo de programación lineal entera y un Greedy Randomized Adaptative Search Process (*GRASP*). Para la creación de soluciones del algoritmo GRASP se implementaron dos heurísticas Demanda satisfecha y Demanda satisfecha por distancia. En la primera, la construcción de soluciones prioriza, en los primeros pasos, los nodos que tengan mayor producción, satisfaciendo rápidamente la demanda solicitada. En los últimos pasos la heurística se enfoca en seleccionar aquellos nodos que tienen una producción que se ajuste más a los requerimientos pendientes minimizando la sobre-recolección. En la segunda heurística no solo se buscan los nodos con mayor producción si no también aquellos que se encuentran a una distancia cercana, esto se realizó multiplicando la distancia del nodo al valor de la demanda satisfecha. Durante la fase de post procesamiento del algoritmo GRASP se realizaron combinaciones de cuatro movimientos para obtener distintos resultados. El movimiento de

reparación realiza un borrado de nodos en la solución, buscando eliminar cualquier excedente en la recolección, sólo es realizado si mejora la solución. El movimiento de intercambio interno evalúa el mejor cambio encontrado de nodos en cada una de las rutas.El movimiento sólo es realizado si mejora la solución. El movimiento de intercambio externo busca realizar el mejor cambio externo posible. Se considera un nodo externo aquel que no pertenece a la solución actual. Solo es realizado si mejora la solución. El movimiento insertar nodo, ingresa a la solución el nodo externo que menos empeore la solución, con este movimiento se busca diversificar la solución. Durante este trabajo se concluyó que el método *GRASP* entrega mejores resultados que el modelo lineal en las instancias de mayor complejidad, análogamente las instancias de menor complejidad fueron resueltas con mejores resultados por el modelo lineal.

Capítulo 3

Propuesta

Como se mencionó en el análisis del estado del arte, si bien se han propuesto algoritmos genéticos en problemas de *PCVRP* obteniendo buenos resultados, no se han aplicado a situaciones de recolección de leche con costos de transporte y sobre demanda del producto. Considerando esto se propone, implementa y evalúa un algoritmo genético para este problema. A continuación se presentan las principales componentes de la propuesta de solución. Se explica la representación de las soluciones y la función de evaluación. A continuación se muestra la estructura general del algoritmo genético y se detallan sus principales procesos.

3.1. Representación

La representación del problema consta de listas que indican las secuencias de granjas visitadas en el recorrido de cada camión. Cada camión empieza y termina en la planta procesadora. En la figura 3.1 se muestra el recorrido de dos camiones. En la figura, el nodo en forma de diamante representa a la planta procesadora, mientras que los nodos circulares representan las granjas del problema. En el ejemplo de solución que representa la figura 3.1, el $Camion_1$ comenzó su recorrido en la planta procesadora para luego visitar las granjas 3, 8, 11 y terminar su recorrido en la planta. El $Camion_2$ tuvo un recorrido que comenzó en la planta procesadora, visito los nodos 2, 7, 6 y acabó el recorrido volviendo a la planta.

$Camion_1$	0	$\Rightarrow 3$	$\Rightarrow 8$	$\Rightarrow 11$	0
$Camion_2$	0	$\Rightarrow 2$	$\Rightarrow 7$	$\Rightarrow 6$	0

Cuadro 3.1: Representación del recorrido de dos camiones.

3.2. Función de evaluación

La función de evaluación se enfoca en minimizar los costos de transporte y recolección de la leche como es mostrado a continuación:

$$\text{Mín} \qquad C \sum_{k \in K} \sum_{(i,j) \in A} d_{ij} x_{ij}^k + \sum_{k \in K} \sum_{i \in N} L_i q_i y_i^k \tag{3.1}$$

Se considera el costo de transporte como los kilometros recorridos por los camiones multiplicado por el costo del kilometro recorrido C. Se considera el costo de recolección como los litros de leche recolectado multiplicado por el costo de leche recolectada L_i.

Para ilustrar la función de evaluación se propone el siguente ejemplo considerando la representación del cuadro 3.1 en donde se calcula el costo de ambos camiones (La información utilizada para calcular el costo de los camiones fue la misma información del caso estudiado en este documento). En este ejemplo se busca recolectar 30000 litros de leche con dos camiones de capacidad de 22000 litros de leche y 14000 litros de leche.

Analizando el recorrido del $Camion_1$, el cual posee una capacidad de recolección de 22000 litros de leche:

- Se inicia el recorrido desde la fabrica (Nodo 0)
- Luego se visita la granja 3, la cual aporta una capacidad de 9315 litros de leche y esta a una distancia de 134 [Km] (Obteniendo un total de 9315 litros recolectados y 134 [Km] recorridos).
- Luego se visita la granja 8, la cual aporta una capacidad de 2192 litros de leche y esta a una distancia de 24 [Km] (Obteniendo un total de 11507 litros recolectados y 158

[Km] recorridos).

- Luego se visita la granja 11, la cual aporta una capacidad de 7671 litros de leche y esta a una distancia de 38 [Km] (Obteniendo un total de 19178 litros recolectados y 196 [Km] recorridos).
- Por ultimo se vuelve a la fabrica (Nodo 0), sumando una distancia recorrida de 94 [Km] (Obteniendo un total de 19178 litros recolectados y 290 [Km] recorridos).

Posteriormente se analiza el recorrido del $Camion_2$, el cual posee una capacidad de recolección de 14000 litros de leche:

- Se inicia el recorrido desde la fabrica (Nodo 0)
- Luego se visita la granja 2, la cual aporta una capacidad de 2658 litros de leche y esta a una distancia de 132 [Km] (Obteniendo un total de 2658 litros recolectados y 132 [Km] recorridos).
- Luego se visita la granja 7, la cual aporta una capacidad de 3288 litros de leche y esta a una distancia de 29 [Km] (Obteniendo un total de 5946 litros recolectados y 161 [Km] recorridos).
- Luego se visita la granja 6, la cual aporta una capacidad de 5479 litros de leche y esta a una distancia de 21 [Km] (Obteniendo un total de 11425 litros recolectados y 182 [Km] recorridos).
- Por ultimo se vuelve a la fabrica (Nodo 0), sumando una distancia recorrida de 127 [Km] (Obteniendo un total de 11425 litros recolectados y 309 [Km] recorridos).

Considerando ambos camiones, se obtiene un total de 30603 litros de leche recolectada y 599 [Km] recorridos. Considerando que el precio de la leche es de un total de 1,99 CLP por litro, y costo de la distancia recorrida esta fijado a 220 CLP por kilometro recorrido entonces el valor final resulta en:

$$220\ \frac{CLP}{[Km]} \times 599[Km] = 131780\ CLP$$
$$1{,}99\ \frac{CLP}{[L]} \times 30603[L] = 60899,97\ CLP$$
$$Costo_{final} = 192679,97\ CLP$$

3.3. Algoritmo Genético

Un algoritmo genético imita el proceso de evolución y selección natural de los seres vivos. Trabaja con una población de soluciones, creada en una fase de construcción, las cuales pasan a través de etapas de transformación (cruzamiento y mutación), elitismo y selección de manera repetitiva generación tras generación. A medida que transcurre la ejecución del algoritmo la población de soluciones va mejorando intentando alcanzar la solución óptima. La mejora del algoritmo se logra generando una nueva población de soluciones, más cercana a la solución óptima, en base a la información de la última población generada, este proceso se repite hasta que se cumpla una condición de termino.

3.3.1. Estructura del algoritmo genético

El algoritmo genético propuesto se divide en cuatro fases como principales: construcción, selección, transformación y elitismo. La estructura general del algoritmo genético propuesto se muestra en el pseudocódigo del algoritmo 1.

El algoritmo comienza con la fase de construcción como se muestra en la línea 3. La fase de construcción es la encargada de crear la primera población de soluciones a trabajar. A continuación la fase de elitismo, en la línea 5 se encarga de reservar y re-introducir las mejores soluciones de la antigua población en la nueva población generada. A continuación se realiza la fase de selección, como se muestra en la línea 6. La fase de selección es la encargada se elegir las soluciones padres que formarán parte del proceso de transformación. A

Algorithm 1 Estructura general del Algoritmo Genético

```
1:  procedure ALGORITMO GENÉTICO
2:      i=0;
3:      Soluciones=Fase de Construcción(ArmadoInicial, TamanoInicial);
4:      while i CantidadIteraciones; do
5:          Fase de Elitisimo(Soluciones, TamanoElitismo, BoolBusqueda);
6:          Fase de Selección(Soluciones, BoolTorneo2);
7:          Fase de Transformación(Soluciones,ProbCruz, ProbMutacion);
8:          flag++;
9:      MejorSol=ObtenerMejorSol(Soluciones);
10:     return(MejorSol);
```

continuación en la línea 7. La fase de transformación realiza las operaciones de cruzamiento y mutación. El cruzamiento realiza la combinación de dos soluciones para encontrar una nueva solución. La mutación por su parte realiza cambios aleatorios en una solución.

A continuación se detalla cada una de las fases del acercamiento propuesto.

3.3.2. Fase de construcción

La fase de construcción realiza el armado de las soluciones iniciales. Esto implica crear un conjunto de *n* soluciones factibles, siendo *n* el tamaño de la población inicial designado en el algoritmo. Para la construcción de soluciones se considera un algoritmo *Greedy* que, como se muestra en el pseudocódigo del algoritmo 2, comenzando desde un punto inicial, en este caso la planta procesadora, selecciona el siguiente componente a incorporar en la solución basado en un criterio heurístico.

Para la construcción de soluciones iniciales de la propuesta se consideraron cinco esquemas que utilizan heurísticas de selección diferentes. Dichas heurísticas se listan a continuación:

- Armado aleatorio.
- Se realiza un greedy prefiriendo la distancia más pequeña al siguiente nodo.

Algorithm 2 Estructura general del Algoritmo Greedy

```
1: procedure GREEDY
2:     Solucion=0;
3:     Agregar Planta a Solucion;
4:     do
5:         MejorNodo=Seleccionar mejor nodo (Heuristica);
6:         Agregar MejorNodo a Solucion;
7:     while Solucion es infactible
8:     Agregar Planta a Solucion;
9:     return(Solucion);
```

- Se realiza un greedy prefiriendo la mayor cantidad de leche a recolectar.
- Se realiza un greedy prefiriendo el nodo que presente menor costo posible.
- Se realiza una operación distancia/leche para encontrar el próximo nodo.

A continuación se explica en detalle cada propuesta heurística de armado de soluciones.

Armado aleatorio – Para el armado aleatorio, se eligen nodos de manera aleatoria, ingresándolas en las rutas de un camión al azar, este proceso se repite hasta que la producción requerida por la planta sea alcanzada. Se implementa esta opción de manera de generar rutas totalmente aleatorias que sean factibles. La diversidad inicial es un aspecto fundamental de los algoritmos genéticos, por lo tanto esta opción puede ser la que mejor cumpla con esta característica en las poblaciones generadas.Este proceso se repite hasta que la cantidad requerida de leche P sea alcanzada.

La figura 3.1 muestra un ejemplo del esquema de armado aleatorio. En este ejemplo, la planta P, representada como un diamante, requiere una cuota de 100 litros de leche. Las 6 granja del problema están representadas con círculos, y sus producciones se muestran arriba de cada nodo. Se considera un camión con capacidad de 150 litros. El vehículo construye una ruta, representada por las flechas rojas, que recoge la leche de las granjas 4,2 y 6. En este ejemplo la elección de granjas es totalmente aleatoria. El proceso de construcción en cada paso eligió

una granja no visitada aleatoriamente. El proceso se detiene una vez que la cuota de leche requerida por la planta es alcanzada.

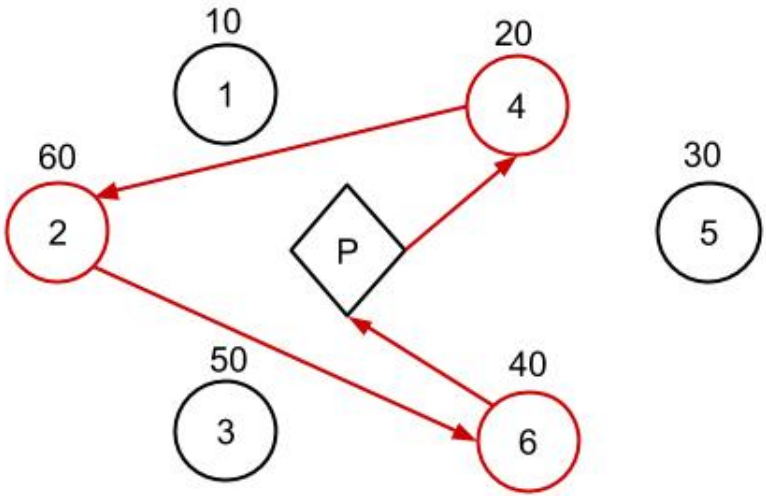


Figura 3.1: Ejemplo de un armado aleatorio.

Armado greedy basado en distancia – Para el segundo armado se busca, en cada paso, el nodo más cercano a la posición actual de un camión al azar. Este nodo es entonces incorporado a la ruta. Este proceso se repite hasta que la cuota de leche requerida por la planta sea alcanzada.

En la figura 3.2 se muestra un esquema representando el armado greedy basado en distancia. En este ejemplo, la planta P, representada por un diamante, requiere una cuota de 100 litros de leche. Usando un camión de capacidad 150 litros. El construye una ruta, mostradas por las flechas rojas del esquema, que recoge la leche de las granjas 4,5,6 y 3. Para la elección de granjas siempre se prioriza el nodo más cercano a la posición del camión, esto se calcula al principio de cada paso.

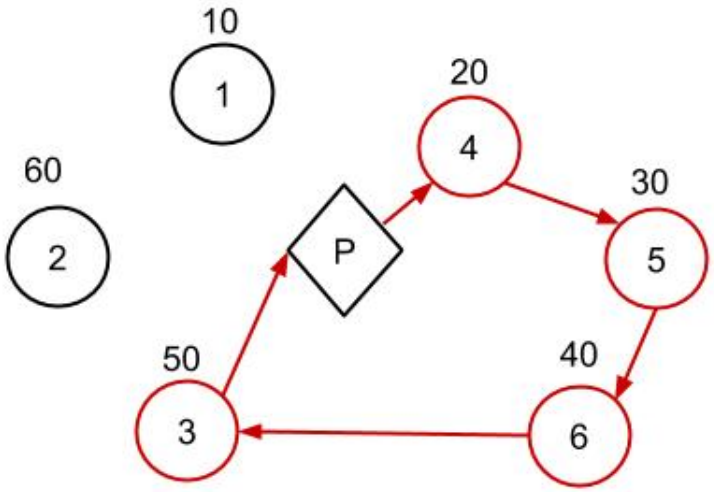


Figura 3.2: Ejemplo de un armado greedy basado en distancia.

Armado greedy basado en recolección – Para el tercer armado se selecciona en cada paso el nodo con mayor cantidad de leche disponible. Se utiliza este criterio promoviendo la satisfacción de la restricción de cuota de leche requerida por el problema. Este nodo se ingresa a la ruta del camión que tenga mayor capacidad disponible, siempre y cuando, la capacidad disponible sea mayor a la cantidad de leche ingresada. Este proceso se repite hasta que la cantidad requerida de leche sea alcanzada.

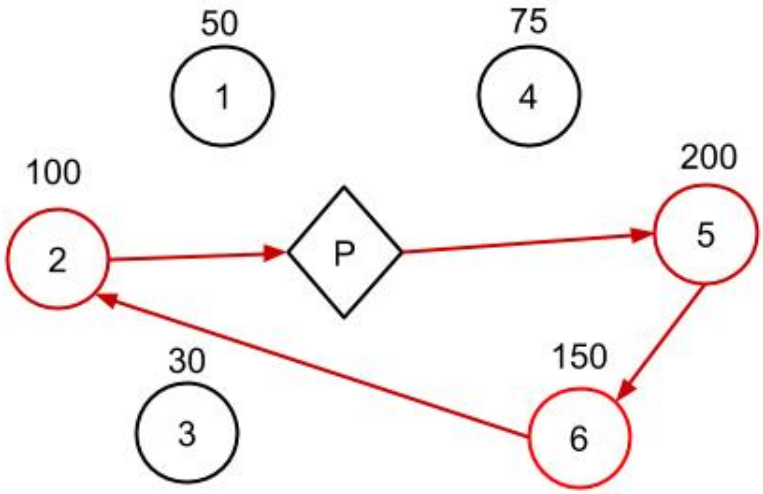


Figura 3.3: Ejemplo de un armado greedy basado en recolección.

En la figura 3.3 se muestra un ejemplo del esquema greedy basado en recolección. En este

ejemplo, la planta P, representada por la figura de un diamante, requiere una cuota de 400 litros de leche. En este caso se construye una ruta considerando la disponibilidad de un camión con capacidad de 500 litros. El vehículo construye una ruta, representada por la línea roja del esquema, que recoge la leche de las granjas 5,6 y 2. Para la elección de granjas siempre se prioriza el nodo que tenga mayor cantidad del producto a recolectar. Así, la primera elección claramente es la granja 5 que posee la mayor producción. Luego la granja 6 que posee la siguiente mayor producción de las granjas remanentes. Finalmente se elige visitar la granja 2 que posee la siguiente mayor producción. Una vez visitada la granja 2 se cumple la cuoya requerida por la planta procesadora por lo que los camiones pueden volver a la planta.

Armado greedy basado en costos de viajes – Para el cuarto armado, al inicio de la fase, se construye una matriz que calcula los costos de viajar entre cada par de nodos NxN. Luego se elige un camión aleatorio y se busca en la matriz el nodo con menor costo a ingresar, este proceso se repite hasta que la cantidad requerida de leche P sea alcanzada. La idea de este esquema es fomentar la elección de los tramos más cortos entre granjas generando así configuraciones de recorridos tipo clusters. Este tipo de configuraciones resulta interesantes pues, a pesar de poseer distancias importantes hacia/desde la base, generan secuencias de visitas poco costosas. Por otro lado, esta secuencia de visitas no es fácil de generar con las heurísticas típicas del vehicle routing problem.

En la figura 3.4 se muestra un ejemplo del uso del esquema de armado greedy basado en costos de viajes. En este ejemplo, la planta P, representada por un diamante, requiere una cuota de 100 litros de leche. Se considera la disponibilidad de un camión con capacidad de 200 litros. El vehículo construye una ruta, representada por la secuencia de flechas rojas que se muestra en el esquema, que visita las granjas 3,6 y 5. Para la elección de granjas se buscan los nodos que presenten un menor costo a la solución.

Armado basado en costos de viajes y producción– Para el quinto armado se elige un camión al azar y se buscan los cuatro nodos con menor costos de viajes. Luego, entre los cuatro nodos elegidos se busca el nodo con mayor producción y se ingresa en el recorrido, este proceso se repite hasta que la cuota de leche requerida por la planta sea alcanzada.

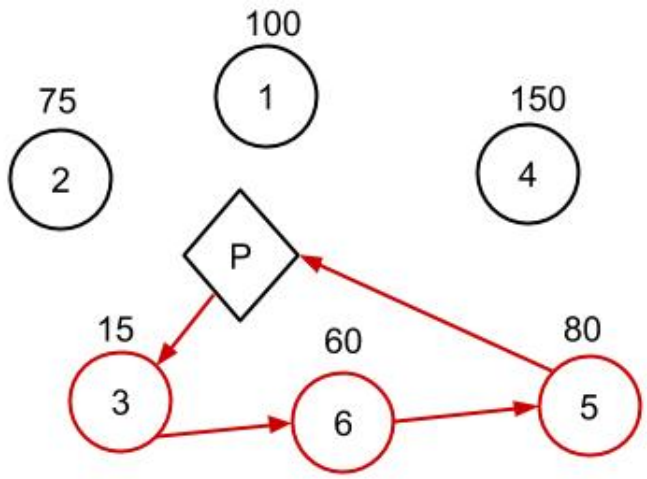


Figura 3.4: Ejemplo de un armado greedy basado en costos de viajes.

En la figura 3.5 se muestra un ejemplo de la aplicación del esquema representando un armado especial. En este caso se buscan los cuatro nodos más cercanos y, entre los elegidos, selecciona la granja con mayor cantidad de leche. En este ejemplo, la planta procesadora P, representada por un diamante, requiere 150 litros de leche. Considerando que se dispone de un camión con capacidad de 300 litros, el vehículo construye una ruta, representada por la secuencia de flechas rojas, recoge la leche de las granjas 1 y 4.

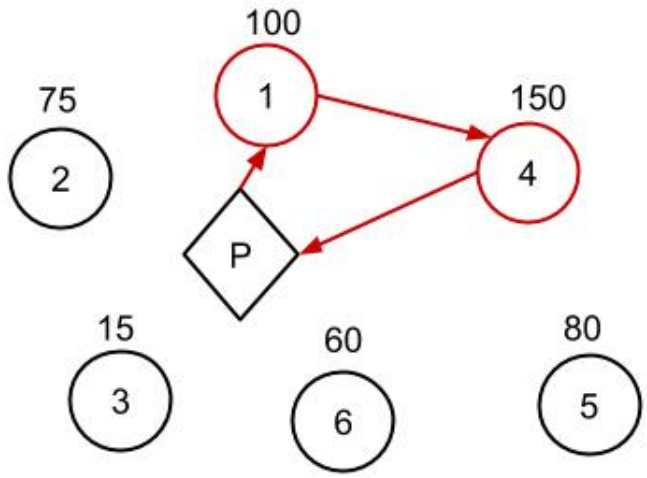


Figura 3.5: Ejemplo de un armado basado en costos de viajes y producción.

Es importante notar que en todos los procesos de armado se busca obtener soluciones factibles, es decir, que la cantidad de leche recolectada sea superior a la cantidad requerida por la planta productora y que la cantidad de leche recolectada por cada camión no supere su capacidad de transporte.

3.3.3. Elitismo

En la fase de elitismo se evalúa la calidad de todas las soluciones obtenidas y se identifican y almacenan las mejores soluciones para que perduren hasta la próxima generación. Durante el proceso de elitismo se implementa una operación de búsqueda local. El proceso de búsqueda local es implementado seleccionando un nodo al azar en cada recorrido de las soluciones elegidas por el elitismo. Este nodo es retirado del recorrido y a continuación reingresado en el lugar óptimo, donde la distancia sea mínima, respecto a la solución completa.

En la figura 3.6 se muestra un ejemplo del proceso de elitismo propuesto. En este caso se consideran dos poblaciones de nueve soluciones cada una. Estas poblaciones representan las poblaciones de padres e hijos. El conjunto de soluciones de la izquierda muestra la población padre y el conjunto de soluciones de la derecha muestra la población de hijos. Cada una de las soluciones muestra un número que representa su calidad. En la población padre se buscan las dos soluciones de mejor calidad, de menor costo en este caso. En el ejemplo estas soluciones están marcadas en verde corresponden a los valores 30000 y 35000. Estas soluciones son agregadas a la población de hijos.

3.3.4. Proceso de selección

Para la fase de selección se implementaron y evaluaron dos operadores: Selección basada en torneo de tamaño 2 y selección aleatoria. A continuación se explican en detalle ambos procedimientos.

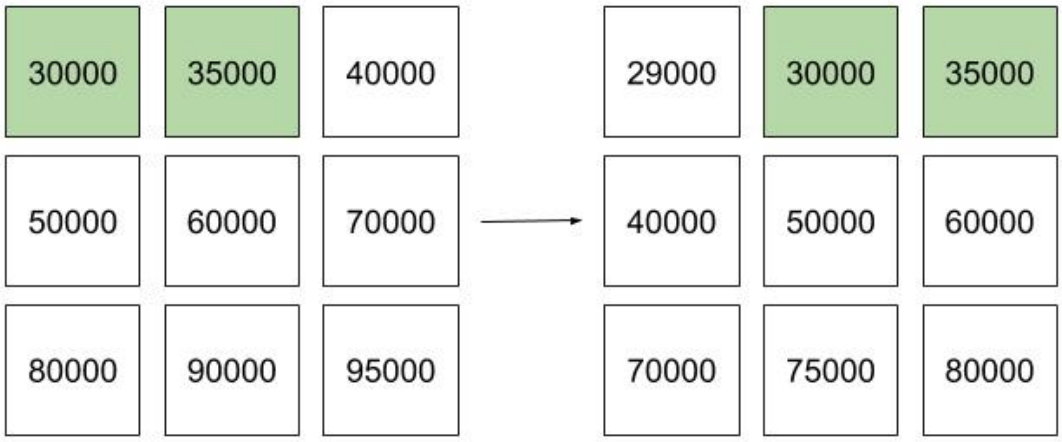


Figura 3.6: Diagrama representativo de la operación realizada en la fase de elitismo.

Selección mediante torneo de tamaño 2

En la fase de selección de padres se implementó el método de torneo de tamaño 2 (torneo-2). El método consiste en elegir dos soluciones con probabilidad de selección uniforme, para luego hacerlas competir y finalmente seleccionar aquella de mejor calidad. El proceso se repite hasta que se seleccione la cantidad de soluciones necesarias para la siguiente población.

En la figura 3.7 se muestran dos poblaciones de nueve soluciones cada una. El conjunto de soluciones de la izquierda muestra la población padre y el conjunto de soluciones de la derecha muestra la población de hijos. Cada una de las soluciones muestra un número que representa su calidad. En la población padre se eligen dos soluciones aleatorias, aquellas de calidad 35000 y 60000. Estas soluciones se muestran en verde en el esquema. Estas dos soluciones compiten y se elige como ganadora aquellas de mejor calidad. En este caso se elige la solución de calidad 35000 pues es la que mejor resuelve el problema estudiado. Este proceso se repite, esta vez se eligen las soluciones marcadas en azul de calidades 80000 y 95000. Las soluciones compiten y gana el torneo en este caso la solución de calidad 80000 que presenta un mejor desempeño para el problema que se resuelve en este caso. El proceso se repite hasta que se consiga seleccionar la cantidad de soluciones necesarias para comenzar el proceso de transformación.

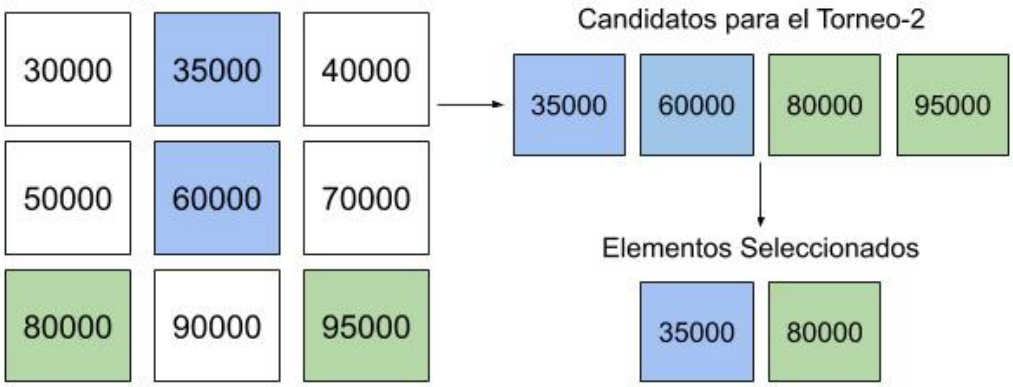


Figura 3.7: Diagrama representativo del esquema de selección de torneo de tamaño 2.

Selección aleatoria

En esta fase también se implementó un operador de selección aleatoria. Este operador elige soluciones, sin considerar su calidad para conformar la población de padres que posteriormente pasará al proceso de transformación. La idea de este procedimiento es evaluar la efectividad de ambos operadores en el proceso de búsqueda general de la propuesta.

En la figura 3.8 se muestran una población de nueve soluciones a la izquierda y dos soluciones a la derecha. Desde la población se eligen dos elementos aleatorios (en azul). Así las soluciones de calidad 70000 y 90000 pasan directamente a la fase de transformación en este caso.

3.3.5. Proceso de transformación

El proceso de transformación considera dos operadores. Un proceso de transformación binario y un proceso de transformación unario. El proceso de transformación binario se implementó como un operador de cruzamiento y e proceso de transformación unario se implementa como un proceso de mutación. Ambos operadores se explican en detalle a continuación.

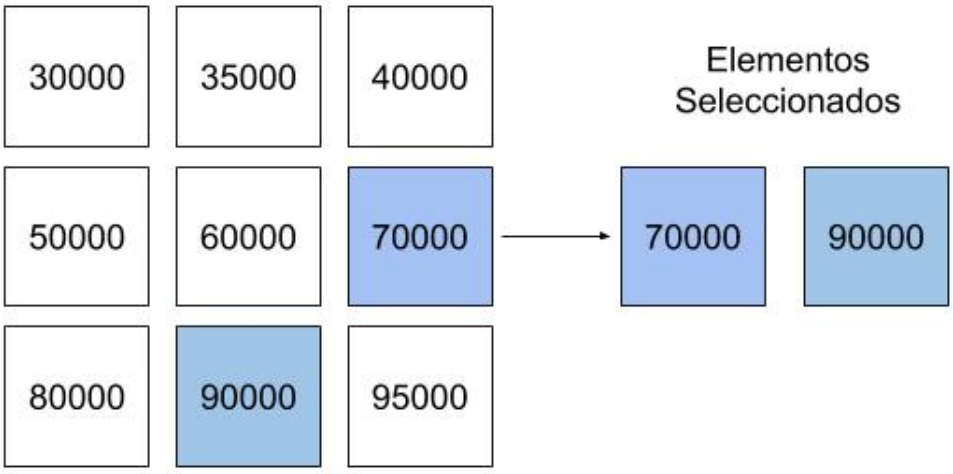


Figura 3.8: Diagrama representativo del esquema de selección aleatoria.

Operador de cruzamiento

El operador de cruzamiento trabaja sobre parejas de soluciones obtenidas en la fase de selección. A partir de cada pareja de padres, referidas como padres, se crea una nueva solución, referida como hijo, basada en la información de ambos padres.

El proceso de cruzamiento consiste en revisar los recorridos de los camiones en los padres y elegir un nodo al azar en ambas soluciones. Una vez seleccionados se reemplaza el nodo del segundo padre en la ruta del primer padre, y la ruta resultante se incorpora en la solución hijo. El proceso se repite para todas las rutas condicionado a la probabilidad del operador de cruzamiento. Una vez terminado el proceso se obtiene una nueva solución.

Es posible que después de haber realizado los cambios la solución se haya vuelto infactible por diferentes razones. Las razones típicas de infactibilidad son: capacidad máxima de leche del camión, más de una visita a una granja, cantidad de leche recolectada total menor a la cuota de la planta procesadora.

La figura 3.9 muestra un ejemplo del proceso de cruzamiento. En este caso se considera una rute con cuatro visitas cada una. $Padre_1$ y $Padre_2$ poseen distintos recorridos. $Padre_1$ visita las granjas 1, 7, 4, y 8 mientras que $Padre_2$ visita las granjas 3, 5, 9 y 2. Basado en la información de ambos padres se crea una nueva ruta ($Hijo$), la cual obtiene información de

ambos padres para armar un recorrido diferente. En este caso, se elige la granja de la segunda posición y el hijo obtiene la secuencia modificada como camino posible para su ruta.

$Padre_1$	0	$\Rightarrow 1$	$\Rightarrow 7$	$\Rightarrow 4$	$\Rightarrow 8$	0
$Padre_2$	0	$\Rightarrow 3$	$\Rightarrow 5$	$\Rightarrow 9$	$\Rightarrow 2$	0

$Hijo$	0	$\Rightarrow 1$	$\Rightarrow 5$	$\Rightarrow 4$	$\Rightarrow 2$	0

Figura 3.9: Ejemplo del uso del operador de cruzamiento.

Operador de mutación

La mutación es un operador unario que toma una solución, elige un nodo al azar entre las granjas no visitadas y la cambia por un nodo al azar de la solución. La operación se realiza únicamente si produce como resultado una solución factible.

La figura 3.10 muestran dos rutas, $Camion_{AntesM}$, siendo el camión con el recorrido original y $Camion_{DespuesM}$, el mismo camión después de haberse realizado la operación de mutación. Es posible observar el cambio realizado por el operador en el tercer elemento del recorrido donde un nodo fue cambiado aleatoriamente (cambiando el tercer elemento visto del nodo 7 al nodo 11).

$Camion_{AntesM}$	0	$\Rightarrow 1$	$\Rightarrow 7$	$\Rightarrow 4$	$\Rightarrow 8$	0

$Camion_{DespuesM}$	0	$\Rightarrow 1$	$\Rightarrow 11$	$\Rightarrow 4$	$\Rightarrow 8$	0

Figura 3.10: Ejemplo del uso del operador de mutación.

Debido a que los cambios en la fase de transformación, más precisamente el cruzamiento, puede resultar en soluciones no factibles, se realiza una fase de reparación posterior a las transformaciones. La fase de transformación se explica a continuación.

Reparación

Durante la fase de reparación se realizan dos fases de reparaciones.

En primer lugar se revisa que la solución no posea nodos repetidos y mantiene la consistencia de los valores de distancia y leche recolectada. Si el valor ingresado en el camión se encuentra presente en otro camión (de la misma solución), el valor previamente existente es retirado dándole prioridad al valor ingresado por el cruzamiento.

La segunda lugar, se revisa si la solución sea factible (que no cumpla la cuota necesaria para satisfacer el problema y/o que los camiones tengan asignado más leche que su capacidad). Para esto se revisa si existen camiones que tengan una carga mayor a su capacidad, si esto ocurre suceden tres pasos:

1. Se revisa si la diferencia de leche excesiva se puede resolver sacando un solo nodo del recorrido de dicho camión.

    a) Se revisa si la diferencia de leche excesiva se puede resolver sacando un solo nodo del recorrido de dicho camión, si es así, se retira el nodo del recorrido cuya capacidad sea menor entre los nodos que, al eliminarlos del recorrido, la cantidad de leche recolectada deje de exceder la capacidad máxima del camión y se finaliza el proceso.

    b) Si retirando un nodo no se puede dejar de exceder la capacidad máxima del camión, se busca el nodo con mayor producción y se retira, luego se vuelve al paso anterior.

2. Se revisa si la cantidad de leche obtenida en la solución cumple con la demanda requerida.

    a) Si la demanda no se cumple, se revisa si existe un único nodo que cumpla con la falta de leche requerida para satisfacer la demanda y que algún camión tenga la capacidad disponible para ingresar dicho nodo luego se ingresa en el camión seleccionado buscando la mejor posición en el recorrido en base a la distancia recorrida y se finaliza el proceso.

b) Si el paso anterior no es posible, se busca el camión con mayor capacidad disponible, luego se busca el nodo con mayor producción que no supere la capacidad restante y se ingresa en el camión seleccionado buscando la mejor posición en el recorrido en base a la distancia recorrida, posteriormente se vuelve al paso anterior.

c) Si todos los nodos restantes son mayores que la capacidad de cualquier camión disponible, se re-distribuyen los viajes de un camión seleccionado a otros camiones, buscando generar la suficiente capacidad para ingresar alguno de los nodos disponibles. Posteriormente se vuelve al paso 2.*a*).

3.3.6. Resumen del capítulo

Para este trabajo se propuso una aproximación de algoritmo genético para el problema de recolección de leche aplicado a una situación real en Chile. Para abarcar un mayor rango de soluciones se implementaron distintas variantes en las fases del algoritmo genético. La fase construcción tiene cinco opciones disponibles para el armado de la solución inicial (listados en la lista 3.3.2), mientras que en la fase de elitismo se implemento una búsqueda local opcional (3.6) y en la fase de selección existe la opción aleatoria 3.8 y Torneo-2 3.7. Por ultimo, en la fase de transformación, se realizan un cruzamiento mezclando la información dos recorridos aleatorios y realizando reparaciones para que la solución se mantenga factible, mientras que la mutación abarco el intercambio de un nodo un recorrido aleatorio por un nodo no visitado siempre y cuando se mantenga una solución factible. En este trabajo se investigara cual combinación de heurísticas producen los mejores resultados en este problema.

Capítulo 4

Experimentos y Resultados

En este capítulo se presentan los experimentos y resultados realizados para validar el algoritmo propuesto para el problema de recolección de leche estudiado en este trabajo. A continuación se describen los objetivos de los experimentos realizados, las instancias de pruebas utilizadas, el entorno de experimentación y el proceso de selección de parámetros. Después de esto, se presentan los resultados y las conclusiones extraíbles de cada uno de los análisis realizados.

4.1. Experimentación

En este capítulo se evalúa el desempeño del algoritmo propuesto. Para esto se han diseñado tres conjuntos de experimentos cuyos objetivos se detallan a continuación.

1. Comprobar la efectividad en el desempeño del algoritmo usando una estrategia de búsqueda local comparándolo con los resultados del algoritmo sin el uso de esta herramienta. La herramienta de búsqueda local consta de intercambiar un nodo aleatoria en cada recorrido con otro nodo del mismo recorrido buscando minimizar la distancia de viaje del vehículo evaluado.

2. Comparar las cinco heurísticas diferentes de selección de componentes en el algoritmo

de generación de soluciones iniciales. Las heurísticas utilizadas fueron, Armado totalmente aleatorio, greedy por leche recolectada, greedy por distancia recorrida, greedy por menor costo a agregar y una heurística creada para este problema al cual llamaremos operación costo/distancia. Cada una de estas heurísticas implementadas tienen distintos enfoques en la resolución del problema, comparar los resultados obtenidos en cada una de ellas brinda información sobre la importancia de la heurística usada en la fase de construcción y como esta decisión impacta en los resultados finales obtenidos.

3. Comparación con el estado del arte. Para esta comparación se consideran las pruebas realizadas utilizando un modelo de programación entera mixta propuesto en [9]. Además, se realizan comparaciones con un algoritmo GRASP de búsqueda local.

4.2. Instancias de Prueba

Los experimentos se realizarán usando un conjunto de casos de estudio obtenidos de una planta procesadora en el sur de Chile. El caso base identificado como *Instancia*1 considera el caso actual de la planta procesadora. La planta procesadora trabaja con un conglomerado de 40 granjas. La distribución de las granjas productoras y sus niveles de producción se muestran en la figura 4.1.

Todas las granjas producen leche del mismo nivel de calidad. Se considera un conjunto variado de productores. Algunas granjas pequeñas producen una cantidad muy baja de leche diariamente y otras más grandes pueden llegar a producir decenas de miles de litros al día. La cantidad de leche producida por estas granjas varía entre los 400 y los 30,000 litros. El caso base estudiado considera tres camiones, uno de 22,000 y dos de 14,000 litros y actualmente se recolectan 35000 litros de leche correspondiente al 17,7 % de la producción total de las 40 granjas.

A partir del caso base se han diseñado ocho escenarios de expansión que consideran inversiones en más camiones que permitirían recolectar una cantidad mayor de leche diariamente.

Para esto, se considera la adquisición de camiones de diferente capacidad. Las capacidades

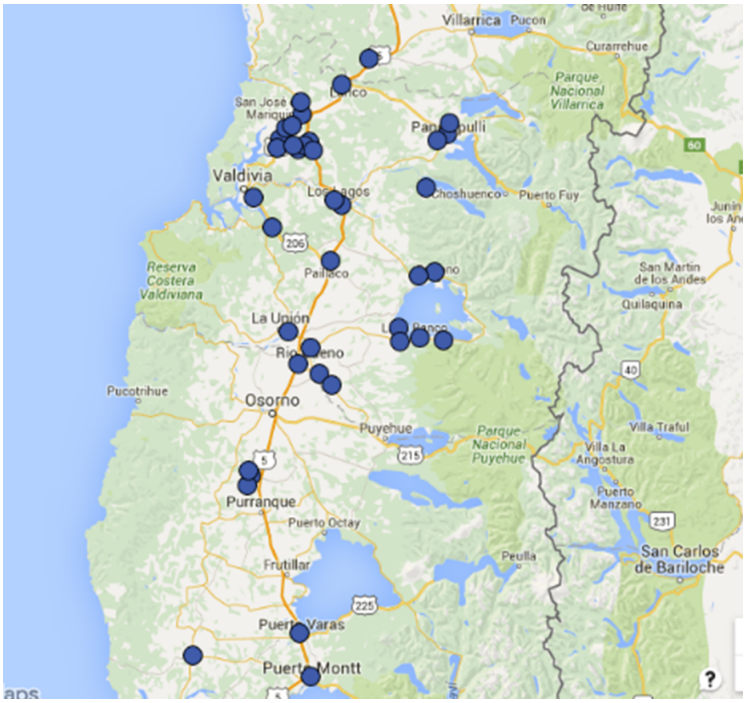


Figura 4.1: Red de productoras de leche en el sur de Chile.

varían entre los 14,000 litros de leche hasta los 30,000 litros. Por otro lado, la cantidad de leche requerida diariamente en las instancias varía entre los 35,000 y 100,000 litros. Los detalles de cada instancia están listados en la tabla 4.1. Para cada una de las instancias se muestra la capacidad total de carga respecto a los camiones que posee, la cantidad de leche requerida por la planta procesadora, la producción total de todas las granjas, la cantidad y capacidad de cada uno de los camiones, el porcentaje de capacidad de los camiones respecto al total de producción, el porcentaje de requerimientos respecto a producción total y la tasa entre requerimientos y capacidad.

Analizando las instancias de la tabla 4.1 se puede prever que algunas instancias tendrán una complejidad mayor que otras. Las instancias 3,6 y 7 presentan mayor tasa de requerimientos y capacidad (Req/Cap). Además, en estas instancias la capacidad total de los camiones es la misma solo cambiando la cantidad de camiones, se presume que la una flota más grande de camiones con mayor capacidad proveerá mayor flexibilidad en la elección de granjas a visitar, minimizando costos considerablemente, por lo que se espera que instancias como la 9, que poseen pequeñas flotas con grandes capacidades, serán más fáciles de resolver.

Instancia	Cap. total	Leche	Prod. total	Camiones	Camiones	%Cap.	%Requ.	Req/Cap
1	50000	35000	197670	3	22,14,14	25.3	17.7	70.0
2	90000	60000	197670	3	30,30,30	45.5	30.4	66.7
3	105000	100000	197670	4	30,30,30,15	53.1	50.6	95.2
4	80000	60000	197670	4	20,20,20,20	40.5	30.4	75.0
5	75000	60000	197670	5	15,15,15,15,15	37.9	30.4	80.0
6	105000	100000	197670	5	30,30,15,15,15	53.1	50.6	95.2
7	105000	100000	197670	6	30,15,15,15,15,15	53.1	50.6	95.2
8	114000	100000	197670	7	22,22,14,14,14,14,14	57.7	50.6	87.7
9	120000	100000	197670	4	30,30,30,30	60.7	50.6	83.3

Cuadro 4.1: Detalles de las instancias utilizadas.

4.2.1. Equipamiento técnico

Los experimentos fueron realizados en un servidor Power Edge R630 con 2 CPUs Intel(R) Xeon(R) E5-2680 v3 @ 2.50GHz, 64 GB de RAM corriendo bajo distribución Ubuntu x64 16.10.

4.2.2. Sintonización de Parámetros

Para sintonizar los parámetros del algoritmo, se hace uso de ParamILS. Propuesto en [8], ParamILS es un método de sintonización automático que utiliza búsqueda local iterativa para encontrar la mejor configuración de parámetros para un algoritmo metaheurístico utilizando como conjunto de entrenamiento un conjunto de problemas de prueba.

Para aplicar ParamILS en el algoritmo propuesto, se definen los dominios deseados de cada uno de los parámetros, un conjunto finito de valores posibles para cada parámetro y un valor inicial para el proceso de sintonización. Dichos valores se presentan en la tabla 4.2. A continuación se explican los parámetros listados en la tabla.

- *ProbMutacion* ($PMUT$): Probabilidad con la que ocurre el operador de Mutación en cada Iteración. La probabilidad de mutación se testea para cada solución analizada.

Parámetro	Valores Disponibles	Valor Inicial
PMUT	{0, 25, 35, 50, 75, 100}	[35]
TPOB	{10, 50, 100, 500, 1000, 2000}	[10]
ELIT	{0, 5, 10, 25, 50, 75, 100}	[5]
PCRUZ	{1, 25, 50, 75, 100}	[100]
ITCRUZ	{0, 10, 50, 100, 500, 1000}	[0]
CINI	{0, 1, 2, 3, 4}	[0]
SEL	{0, 1}	[0]
LS	{0, 1}	[0]

Cuadro 4.2: Parámetros sintonizados por ParamILS. Para cada parámetros se presenta su conjunto de valores posibles y valores iniciales.

- *TamanoPoblacion* (*TPOB*): Cantidad de soluciones con la cual se trabajara el algoritmo genético.
- *TamanoElitismo* (*ELIT*): Porcentaje de las mejores soluciones (*TamanoPoblacion*) en la generación actual que se reserva para la siguiente generación.
- *ProbCruz* (*PCRUZ*): Probabilidad con la que ocurre el operador de Cruzamiento en cada Iteración. La probabilidad de cruzamiento se testea para cada par de soluciones analizadas.
- *IteracionesCantidad* (*ITCRUZ*): Condición de termino, corresponde al número de iteraciones que se realizan antes de terminar el algoritmo.
- *ArmadoInicial* (*CINI*): Este valor designado indica que procedimiento de construcción de soluciones iniciales se utiliza para generar la población inicial. Dependiendo del valor se realiza uno de los armados iniciales definidos con anterioridad.
- *BoolTorneo*2 (*SEL*): Este valor indica qué proceso de selección se utiliza en el algoritmo. Si se realiza torneo-2 (valor=1) o se realiza una selección aleatoria (valor=0).
- *BoolBusqueda* (*LS*): Este valor indica si la fase de elitismo realiza la búsqueda local (Valor=1) o no la realiza(Valor=0).

Para el proceso de sintonización se utiliza como medida de calidad de cada ejecución lo que se conoce como distancia al óptimo (gap). Esta distancia se calcula usando la expresión 4.1, en la que C corresponde a la calidad de la solución obtenida en la ejecución del algoritmo y $C*$ corresponde a la mejor calidad encontrada para dicha instancia en el estado del arte.

$$gap = \frac{C* - C}{C*} \tag{4.1}$$

Luego de ejecutar ParamILS para la configuración descrita, los resultados obtenidos indican que la configuración óptima del algoritmo es la expuesta en el cuadro 4.3.

Parámetro	Valor
PMUT	100
TPOB	2000
ELIT	25
PCRUZ	50
ITCRUZ	1000
CINI	1
SEL	0
LS	0

Cuadro 4.3: Configuración de parámetros entregada por ParamILS

Los valores obtenidos del proceso de sintonización se explican de la siguiente manera:

- ($PMUT$): En el caso de la mutación, una mayor probabilidad de mutación implica una mayor exploración, muy necesario debido a solo trabajar con soluciones factibles lo cual limita la exploración del algoritmo.
- ($TPOB$):Tener poblaciones de mayor tamaño permiten poseer una rango de soluciones más amplio para trabajar, lo cual ayuda a obtener soluciones de mejor calidad a costa de tiempo de ejecución.
- ($ELIT$): Trabajar con un 25 % de elitismo implica retener gran parte de las mejores

soluciones ya obtenidas, lo cual balancea la exploración con la explotación en busca de buenos resultados. Es importante recordar que el elitismo favorece a la explotación.

- ($PCRUZ$): El cruzamiento es el único movimiento que puede realizar cambios no factibles seguidos de un proceso de reparación. Es probable que los movimientos de reparación no sean óptimos para explorar las soluciones, ya que no solo buscan reparar la solución, si no hacerlo de la manera más óptima, por lo que se busca minimizar esta operación a un 50 %.

- ($ITCRUZ$): La cantidad de iteraciones en el proceso de cruzamiento ayuda tanto a la exploración como a la explotación, una mayor cantidad de iteraciones garantiza un mejor resultado que una menor cantidad de estas.

- ($CINI$): El armado inicial de valor 1 implica que se favoreció el armado greedy en base a distancia, siendo la distancia uno de los factores más impactantes al momento de calcular el coste final, enfocarse en este armado inicial resulta una elección lógica para obtener los mejores valores.

- (SEL): El método de $Torneo - 2$ es una heurística de explotación muy fuerte, convergiendo a resultados de manera muy apresurada y limitando los mejores resultados que podría obtener el algoritmo.

- (LS): La búsqueda local ayuda a la explotación cada vez que se realiza una fase de elitismo, considerando que el problema favorece fuertemente la exploración para encontrar mejores soluciones. Es de esperar que se prefiera la no utilización de esta heurística, sin contar que además la no utilización de la búsqueda local ayuda a obtener mejores tiempos.

4.3. Resultados

En esta sección se presentan los resultados obtenidos a partir del proceso de experimentación propuesto. Para cada uno de los análisis se presentan los resultados de calidad y tiempo de cómputo de cada uno de los esquemas analizados usando boxplots.

Un boxplot es un método para graficar grupos de datos numéricos usando cuartiles. Para cada grupo de datos, el boxplot muestra, de izquierda a derecha, el valor mínimo de la muestra, la mediana, los cuartiles y el valor máximo. Observando los boxplots se puede ver si las distribuciones son simétricas, si existen outliers en la muestra, y la distribución de los datos a lo largo de la escala de cuartiles. Cada uno de los boxplots fue construido en base a los resultados de 20 ejecuciones independientes del algoritmo indicado en cada caso.

A continuación se presentaran los resultados de las estrategias de búsqueda local y operadores de selección comparándolos bajo un mismo armado inicial, posteriormente se analizaran los resultados de distintos armados iniciales sin utilizar las heurísticas de búsqueda local ni *torneo* − 2 y se discutirá los resultados obtenidos en ambos análisis.

4.3.1. Estrategias de Búsqueda local y operador de selección

Se evaluaran las heurísticas de búsqueda local y del operador de selección. La heurística de búsqueda local se aplica cada vez que se realiza la fase de elitismo. En el proceso de búsqueda local se selecciona un nodo aleatorio por recorrido y se busca la posición óptima, en cualquier ruta, que implique disminuir la distancia total recorrida.

Las heurísticas del proceso de selección son *torneo* − 2, donde se obtienen dos soluciones al azar de manera aleatoria, y se elige la solución de mejor valor. Este proceso de repite para cada solución que se desee seleccionar. La segunda heurística de selección es la selección aleatoria. La comparación de estos resultados sera utilizada para evaluar cuales son las heurísticas que entregan mejores soluciones y descubrir por que esto ocurre. Las configuraciones de parámetros acá comparadas se resumen en la tabla 4.4.

A continuación se analizaran los resultados obtenidos de las distintas configuraciones realizando dos estudios principales, el análisis de los resultados entre las configuraciones $C1-C4$, que tienen el mismo armado inicial (*Greedy Distancia*) pero varían los métodos de selección y búsqueda local Los detalles se muestran en el cuadro 4.4).

En `Costos Instancia 1` todos los resultados convergen a una única solución, esta es la instancia con menor requerimiento de leche y, por ende, que realiza menos carga en sus

Configuración	Armado Inicial	Torneo-2	Búsqueda Local
C1	Greedy Distancia	No	No
C2	Greedy Distancia	Si	No
C3	Greedy Distancia	No	Si
C4	Greedy Distancia	Si	Si

Cuadro 4.4: Detalles de las configuraciones utilizadas.

camiones. Así, en la mayoría de los casos se llega a un resultado óptimo. Respecto a los `Tiempos Instancia 1` se aprecia que los tiempos de ejecución mantuvieron un patrón entre $C1/C3$ y $C2/C4$. Siendo $C1$ y $C3$ las dos configuraciones que no utilizan la heurística de $Torneo-2$, mientras que $C2$ y $C4$ si la utilizan, sugiriendo que esta heurística agiliza el tiempo de ejecución.

Respecto a la calidad de las soluciones de la instancia 2 se puede verificar como $C1$ entrega los mejores resultados llevando la mayoría de sus soluciones al mismo resultado. Eso sí, con una mayor cantidad de outlayers. Se aprecia que en esta configuración no existe una mayor diversidad de soluciones como sí pasa en las otras tres configuraciones ($C2/C3/C4$). La configuración $C3$ es la que compite más fuertemente contra la configuración propuesta por ParamILS. $C2$ y $C4$ son claramente las que tienen peor desempeño en esta instancia debido a la utilización de $Torneo-2$.

Observando los tiempos de ejecución, no existen mayores variaciones en comparación a la instancia 1. Nuevamente los tiempos de ejecución fueron ligeramente menores (a excepción de $C3$). Esto a pesar de suponer una mayor carga para el algoritmo, considerando que la tasa requerida de leche en comparación a la capacidad de los camiones es mayor en la instancia 1 (Teniendo una tasa de 70.0, mientras que la tasa de la instancia 2 es de 66.7).

En la instancia 3 se observa una situación similar a la instancia anterior. Analizando el comportamiento de costos y tiempos pero se puede apreciar que los costos aumentaron considerablemente, esto es de esperar considerando que la instancia 3 no solo pide una mayor cantidad de leche que en las instancias anteriores, pidiendo una cantidad de 100000 litros

de leche (una gran cantidad en comparación de las instancias 1 y 2 que demandan 35000 y 60000 respectivamente), si no que también es la que tiene una mayor tasa de requerimiento y capacidad alcanzando un valor de 95.2.

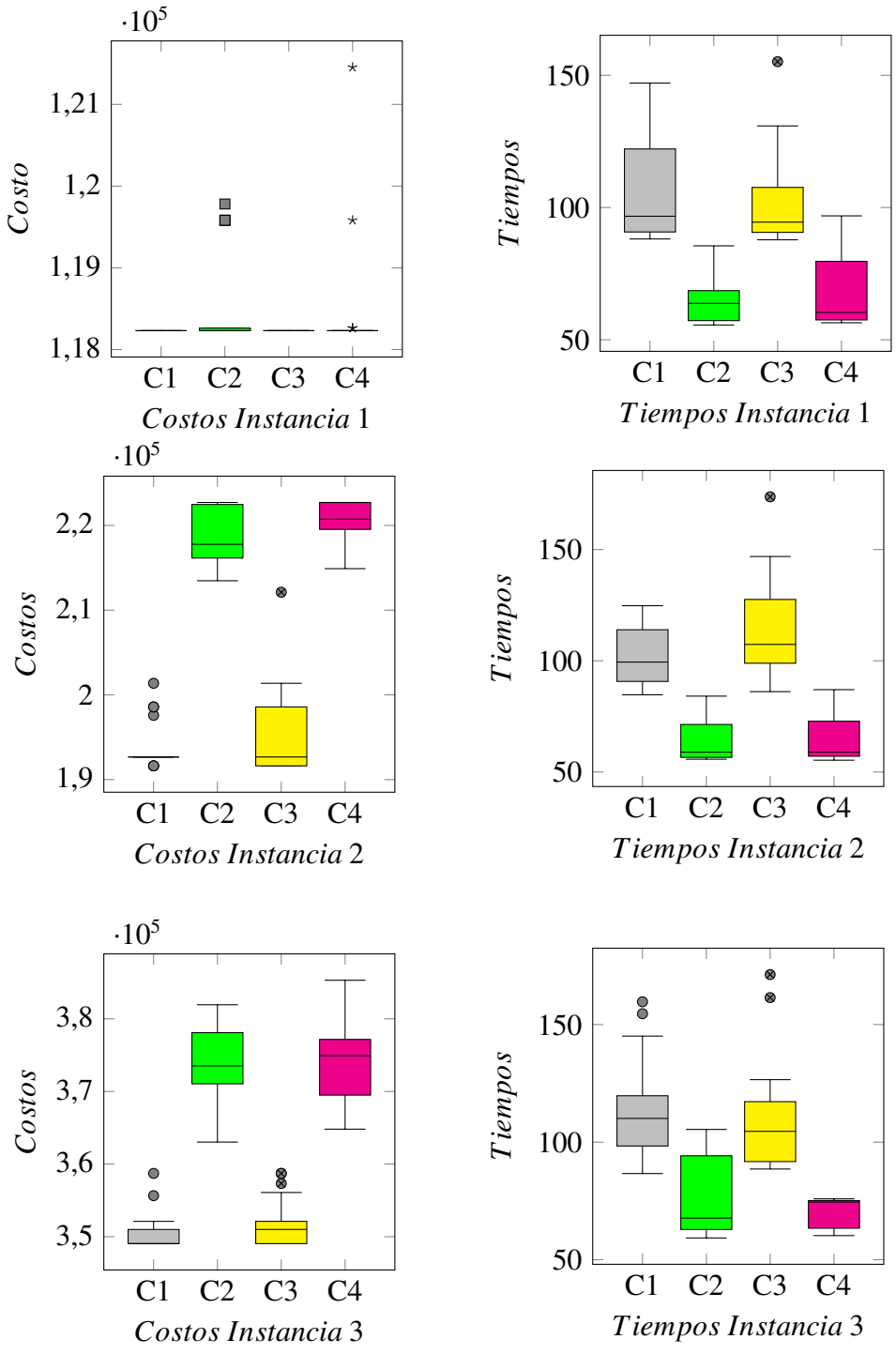


Figura 4.2: Comparación entre los resultados obtenidos para las instancias 1-3 con las configuraciones C1-C4

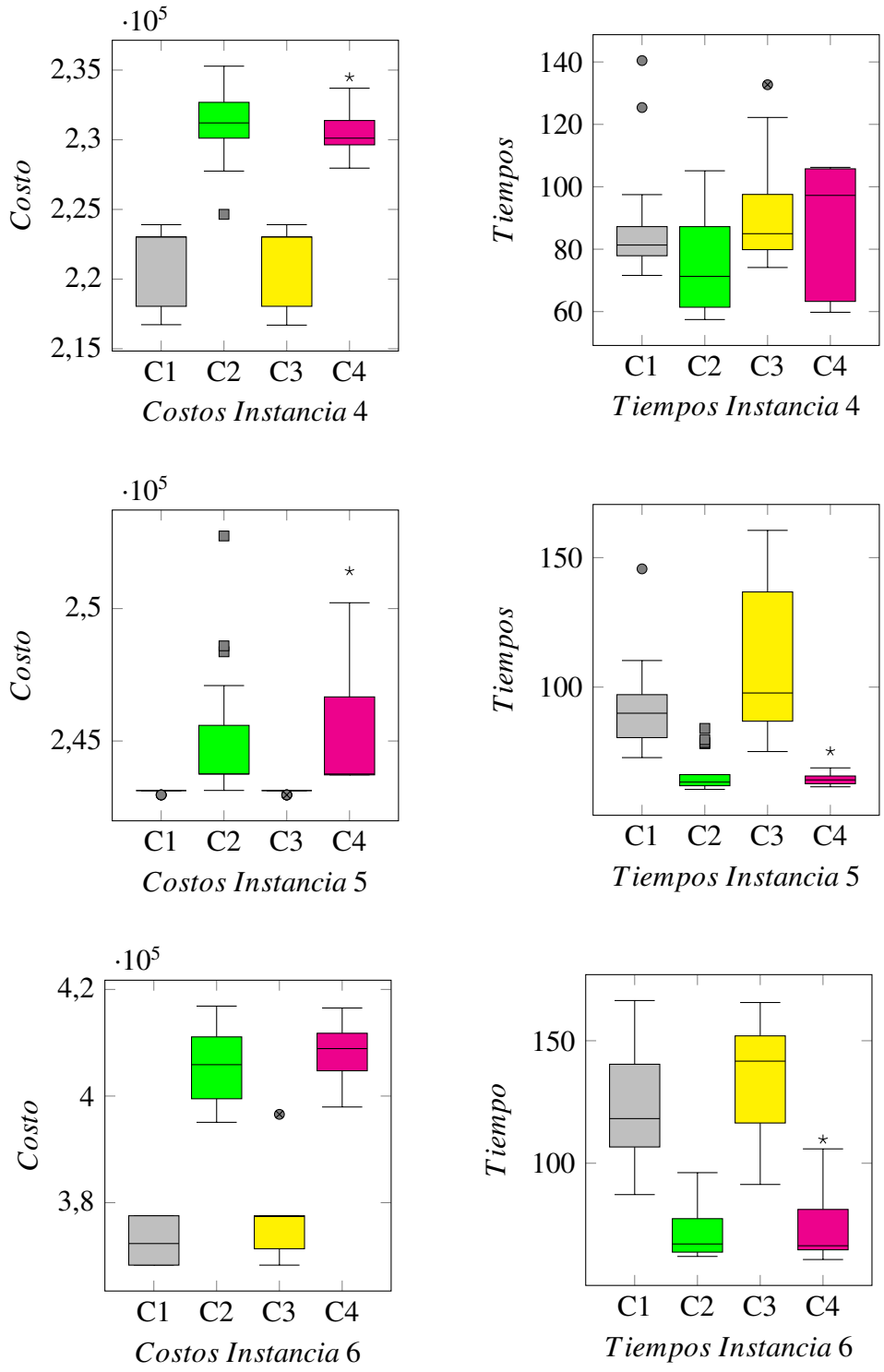


Figura 4.3: Comparación entre los resultados obtenidos para las instancias 4-6 con las configuraciones C1-C4

En los costos de la instancia 4 se puede apreciar que los costos totales bajaron en comparación a la instancia anterior (instancia 3), a pesar de trabajar con la misma cantidad de camiones, se teoriza que esto se debe a que la tasa de requerimientos y capacidad baja a 75.0, implicando que los camiones no están llenos a tope y tienen mayor libertad para elegir las granjas productoras en sus recorridos. Las configuraciones $C1$ y $C3$ siguen entregando resultados parejos y de mejor calidad que las otras configuraciones. Resulta interesante observar que las configuraciones $C2$ y $C4$ que presentan tiempos máximos tan altos como $C1$ y $C3$. Considerando que los tiempos de ejecución están fuertemente ligados a la cantidad de reparaciones realizadas y que la heurística de Torneo-2 favorece fuertemente la explotación, se puede teorizar que las mejores soluciones poseían las mismas granjas pero distribuidas en distinto orden, entonces, al momento de realizar el cruzamiento las soluciones seleccionadas chocaban constantemente y, por ende, necesitaban ser reparadas con mayor frecuencia.

La instancia 5 resulta una de la más interesantes, mostrando calidades que usando $C1$ y $C3$ convergen a una solución rápidamente mientras que $C2$ y $C4$ abarcan una mayor cantidad de peores soluciones. Respecto al tiempo el efecto es totalmente opuesto. Aquí $C2$ y $C4$ prácticamente no muestran variación en sus tiempos de ejecución a diferencia de las otras dos configuraciones. Como se puede apreciar en la Tabla 4.1, esta es la primera instancia analizada con una flota homogénea de vehículos, además, todos los camiones utilizados son 15 mil litros de capacidad, lo cual es considerado una de las menores capacidades dentro de las posibilidades, lo que significa menor libertad para visitar las granjas de mayor producción. Es curioso notar que $C2$ y $C4$, aquellas configuraciones que se centran en una mayor explotación, sean aquellas que entreguen una mayor variedad de soluciones en respecto a los costos. Esto puede ser debido a que a pesar de converger más rápido a una única solución, explicando el menor tiempo de computo, las poblaciones de soluciones nunca se acercan al óptimo, por lo que en cada experimento llegan a un resultado diferente.

Al observar los costos de la instancia 6 se puede apreciar que se volvió a la forma más común de los resultados obtenidos, con $C1$ y $C3$, nuevamente, entregando los mejores resultados y tomando una mayor cantidad de tiempo. El costo y el tiempo de ejecución subieron considerablemente, pero es de esperarse considerando que la instancia 6 posee una tasa de requerimientos y capacidad muy alta, similar a la instancia 3 (Tabla 4.1).

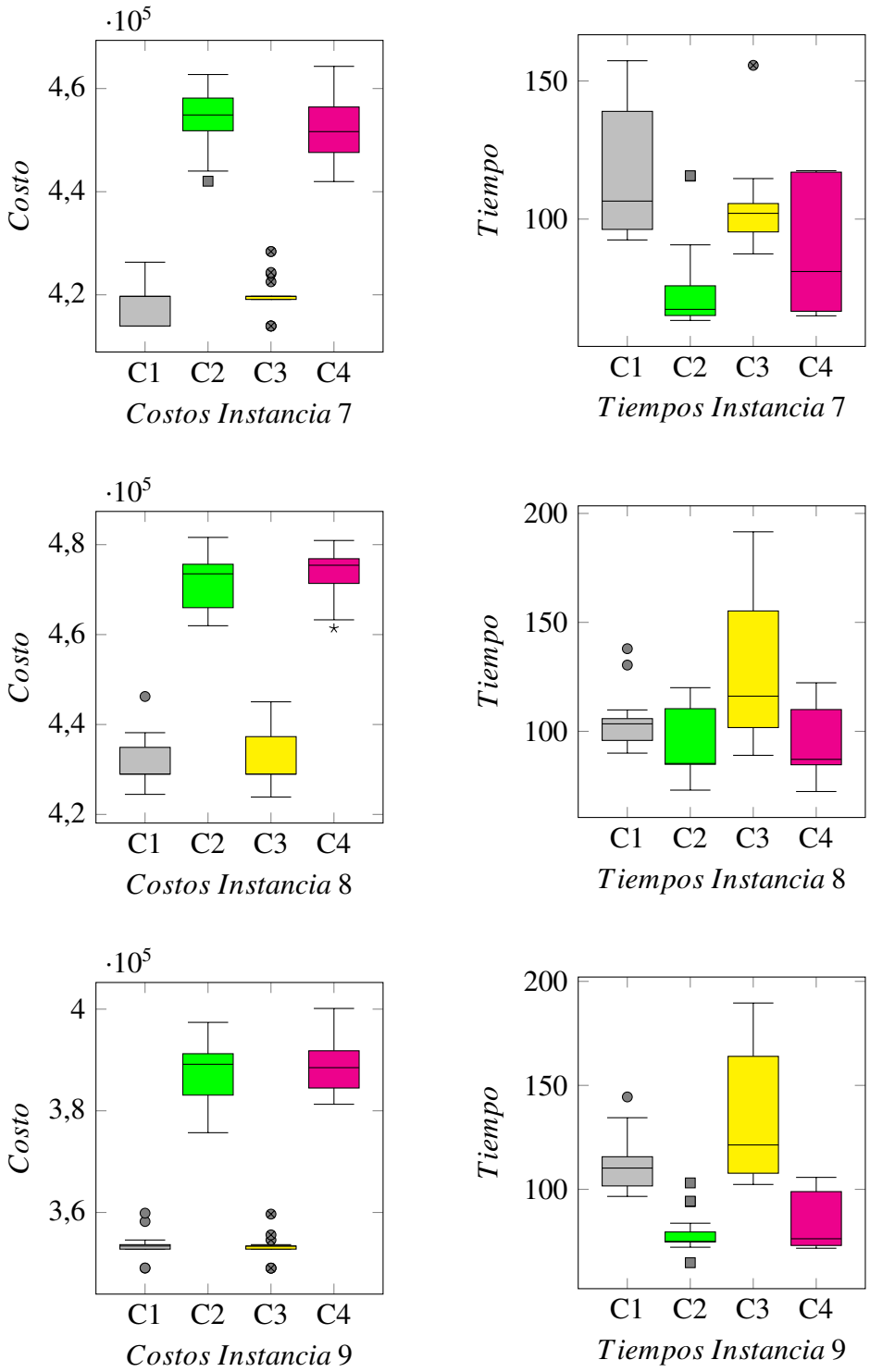


Figura 4.4: Comparación entre los resultados obtenidos para las instancias 7-9 con las configuraciones C1-C4

Las instancias 7, 8 y 9 son las más complejas por resolver debido a que se consideran las mayores cantidades de leche a recolectar y las flotas de camiones transportadores de mayor tamaño. Esto se puede apreciar en el valor elevado del costo en las tres instancias. Es importante observar que los resultados de los costos de las tres instancias siguen el mismo patrón presentado con anterioridad. $C1$ y $C3$ son las configuraciones que entregan mejores resultados. Se puede apreciar los tiempos de las instancias 7 y 8 que $C2$ y $C4$ entregan tiempos de menor calidad, en instancias mucho más complejas con mayor cantidad de camiones parece haber casos en los cuales Torneo-2 resulta en grandes tiempos de calculo (Este análisis se refiere específicamente a$C4$ en `Tiempos Instancia 7` y $C3$ en `Tiempos Instancia 8`).

Respecto a los tiempos de ejecución se observa que en la instancia 9 se vuelve a la tendencia que indica que $C2$ y $C4$ presentan mejores tiempos que las otras dos configuraciones a pesar de ser la instancia que requiere una mayor cantidad de leche en comparación a las otras. Indicando que la cantidad de leche, por si misma, no presenta una dificultad en para el algoritmo, si no más bien es la tasa de requerimientos y capacidad, junto con la cantidad de camiones a trabajar.

Realizando un análisis general en todas las instancias anteriores (Figuras 4.2, 4.3 y 4.4), las configuraciones $C2$ y $C4$ presentan los menores tiempos de ejecución, pero también muestran los peores resultados de calidad. La característica que diferencia estas configuraciones es que se utiliza torneo-2 para la fase de selección. Esto podría implicar que este tipo de selección converge más rápidamente a una solución, explicando que la cantidad de tiempo para estos experimentos sea menor, pero muestren resultados de peor calidad. Esto debido a que la exploración del problema es reducida en favor de una mayor explotación.

Volviendo a comparar las mismas parejas de configuraciones ($C1$ y $C3$, $C2$ y $C4$) en toda esta sección (Figuras 4.2, 4.3 y 4.4) es posible notar que $C1$ y $C2$, tienden a tener mejores resultados. Esto probablemente debido a que estas no utilizan búsqueda local. Si bien la búsqueda local es una buena herramienta para optimizar resultados, ocurre un problema similar al momento de utilizar torneo-2, se pierde exploración.

Además otro problema que se presenta con esta herramienta es que, debido a las características de las instancias utilizadas para estas pruebas, los recorridos resultantes tienden

a considerar pocas granjas. Esto ya que existen granjas productoras que proveen una gran cantidad de leche a recolectar y a pesar de presentarse a distancias lejanas, minimizar la cantidad de viajes a distintos nodos tiende a ser beneficioso para el problema en el largo plazo. Así, ya que las distancias de viaje son las mismas entre el viaje de ida y de regreso, no existe beneficio alguno en tratar de reordenar los nodos en base a la distancia recorrida para recorridos con tan pocas paradas. A pesar de que la media de $C3$ tienda a ser de peor calidad que la media de $C1$, las mejores soluciones de la tercera configuración son siempre tan buenas, o mejores, que las de la primera, implicando que, a pesar de que el promedio de las soluciones no sea de mejor calidad, la explotación logra encontrar buenos resultados, por lo que una mezcla entre explotación y exploración puede ser la mejor opción para obtener buenos resultados en estos casos.

Con respecto a las instancias, es notorio que los tiempos de ejecución, van aumentando a medida que se van resolviendo instancias más complejas. De acuerdo a la tabla 4.1 esto puede ser debido a que las instancias van demandando mayor cantidad de leche con una cantidad similar de camiones, aumentando así la tasa de requerimientos y capacidad. Además se utilizan más camiones para recolectar dicha cantidad. Es interesante comparar las instancias 3 y 6, ya que la empresa procesadora solicita la misma cantidad de leche en ambas y las capacidades de los camiones son similares, diferenciándose solo en la cantidad de camiones. Notoriamente la instancia 3 presenta mejores resultados que la instancia 6 sugiriendo que instancias con menor cantidad de camiones pueden realizar recorridos más óptimos y/o visitar menos granjas con mayor capacidad de producción. Este último punto se refuerza al comparar las instancias 7 y 9, en donde la instancia 9 posee menor capacidad de recolección que la instancia 7, pero posee exclusivamente camiones de 30 mil litros de capacidad, mayor capacidad de lo que presentan la mayoría de los camiones de la instancia 7. Así, la instancia 9 presenta resultados mucho mejores que los presentados por la instancia 7. Por ultimo, se puede observar que la instancia 8, la que posee una mayor cantidad de camiones y una flota más homogénea, pero con menor capacidad en sus camiones, es la que obtiene peores resultados.

Por lo general, la configuración $C3$ tiende a ser más lenta que $C1$, esto es de esperarse ya que $C3$ realiza más operaciones durante la fase de elitismo. A pesar de esto, resulta curioso notar

que $C3$ efectivamente posee menores tiempos en las instancias 1 y 3. Si bien la aleatoriedad podría ser un factor, se noto que tanto la primera como tercera instancia de prueba poseen una característica común, estas son las instancias que poseen una capacidad de recolección total más cercanas a la cantidad de demanda total presentada. Durante todo el experimento se busca trabajar con soluciones factibles, esto no presenta mayores inconvenientes en la mayoría de las instancias ya que la capacidad de recolección tiende a ser bastante mayor a la demanda total, dejando lugar para poder realizar una mayor exploración en cada configuración. Por otro lado, en las instancias 1 y 3 al tener menor capacidad extra de carga en los camiones, la explotación entrega resultados de mejor calidad y un menor tiempo.

4.3.2. Estrategias de construcción de soluciones

En esta sección se evaluaran las heurísticas de la fase de construcción, existiendo cinco tipos de armado inicial: greedy basado en distancia, aleatorio, greedy basado en producción, greedy basado en costos y basado en distancia y leche. En este experimento se comparan los cinco esquemas de construcción para evaluar qué heurística aporta mejores soluciones en este problema. Cada heurística en esta sección tiene un enfoque diferente, priorizando más, o menos, atributos de exploración o explotación. A partir de este análisis no solo se podrá verificar cual es la mejor aproximación para este problema, si no que también se podrá teorizar cuales tendrán mejor éxito en trabajos futuros.

Las configuraciones de parámetros acá comparadas se resumen en la tabla 4.5. En los boxplots se le asignaron colores a cada configuración para que fuese más fácil identificarlos, la configuración $C1$ es de color gris, $C2$ es de color verde, $C3$ es de color amarillo, $C4$ es de color rosado, $C5$ es de color naranjo, $C6$ es de color purpura, $C7$ es de color café y por ultimo $C8$ es de color azul.

Configuración	**Armado Inicial**	**Torneo-2**	**Búsqueda Local**
C1	Greedy Distancia	No	No
C5	Aleatorio	No	No
C6	Greedy Leche	No	No
C7	Greedy Costo	No	No
C8	Distancia/Leche	No	No

Cuadro 4.5: Detalles de las configuraciones utilizadas.

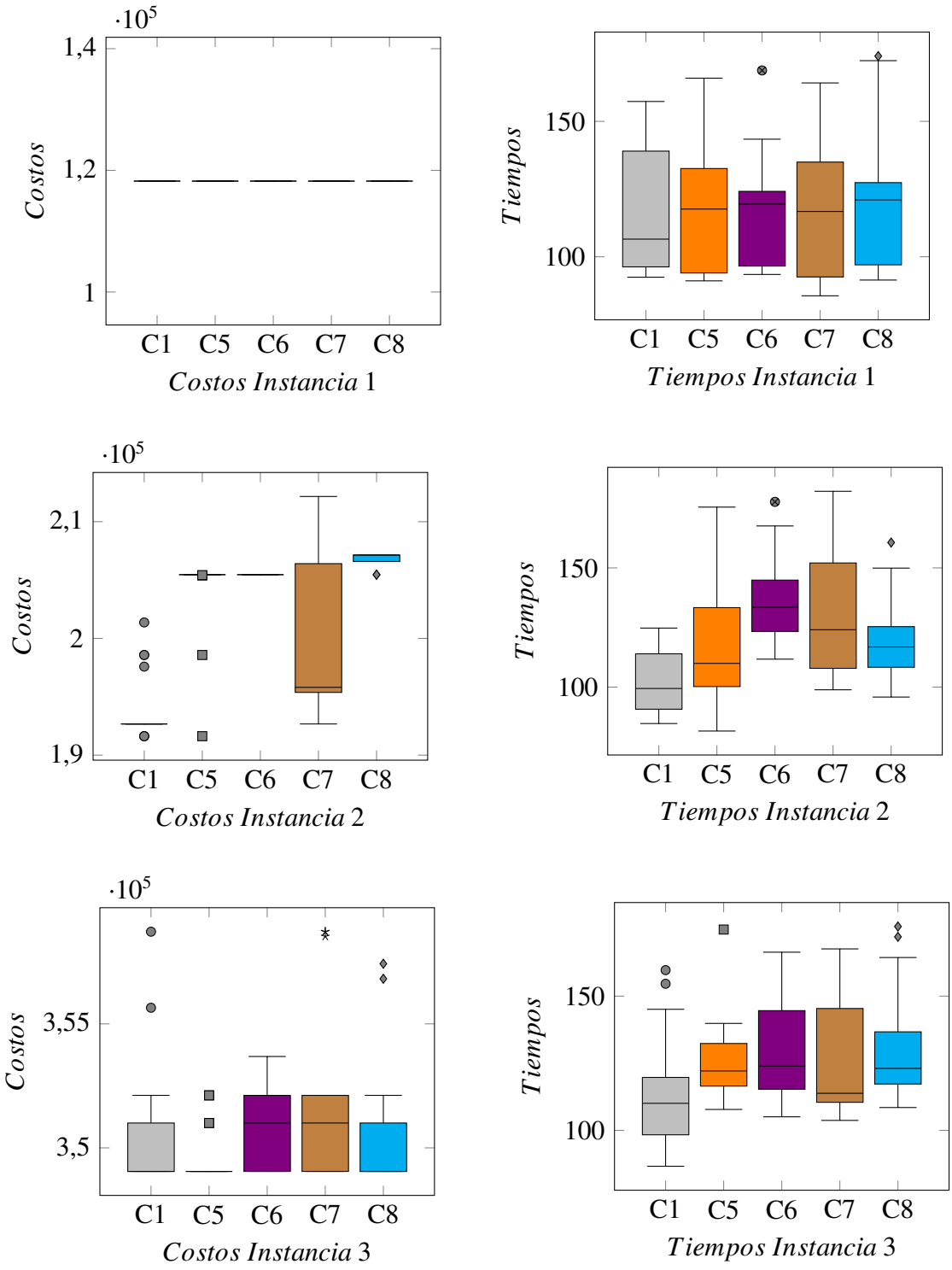


Figura 4.5: Comparación entre los resultados obtenidos para las instancias 1-3 con las configuraciones C1,C5,C6,C7,C8

En la figura 4.5 es posible observar que para la instancia 1 todos los resultados convergen a una única solución, este fenómeno es similar al de la figura 4.2. Siendo la instancia con menor requerimiento de leche y por ende que realiza menos carga en sus camiones es esperable que todas las configuraciones lleguen a un resultado óptimo. Es importante considerar que en estas configuraciones no se encuentra presente la búsqueda local ni el método de selección *torneo* − 2, por lo cual es mucho más probable que las distintas configuraciones abarquen resultados similares. Respecto a los tiempos, se aprecia que en la instancia 1 los tiempos de ejecución fueron relativamente parejos.

Respecto a la instancia 2 se puede verificar como $C1$ entrega los mejores resultados llevando la mayoría de sus soluciones al mismo resultado. Un efecto similar ocurre con $C5$ pero con soluciones de peor calidad. Las configuraciones $C6$ y $C8$ muestran poca variación en sus resultados, a diferencia de $C7$, siendo el armado greedy basado en costos el con mayor variedad de soluciones. En términos de tiempo se observa que las medias de todas las configuraciones permanecen cercanas entre sí, siendo $C1$ la más eficiente entre las configuraciones evaluadas y $C6$ la menos eficiente. Vale notar que en la tabla 4.1 se nota que la instancia 2 es la que muestra menor tasa entre requerimientos y producción, sin mencionar que todos los camiones de recolección tiene una gran capacidad de recolección.

La instancia 3 es la primera instancia que presenta una alta tasa entre requerimientos y producción (Tabla 4.1) esto puede ser apreciado en los elevados costos que se obtienen. Vale notar que la configuración $C5$ entrega resultados consistentes, insinuando que en resultados con alta tasa, donde no existe tanta libertad para exploración de las soluciones mediante las fases transformación. La construcción inicial aleatoria, la cual en si misma beneficia la exploración, posee una ventaja entre sus pares. Los tiempos en la instancia 3 muestran nuevamente una cierta igualdad entre las configuraciones pero, nuevamente, la configuración $C1$ muestra una ligera ventaja sobre sus pares, implicando que la minimización de distancias podría ser clave al momento de realizar algoritmos de búsqueda enfocados en el tiempo de ejecución.

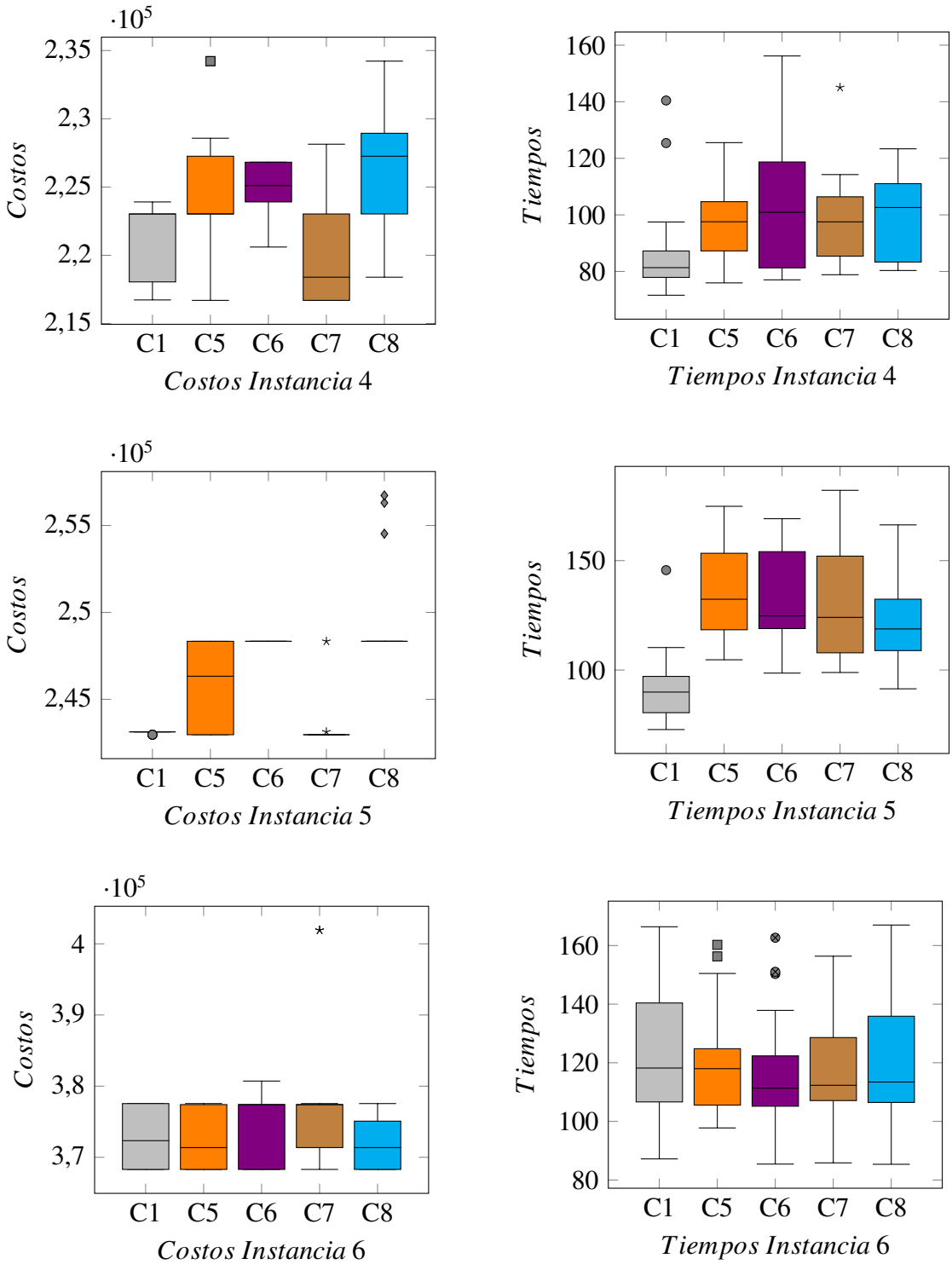


Figura 4.6: Comparación entre los resultados obtenidos para las instancias 4-6 con las configuraciones C1,C5,C6,C7,C8

La instancia 4 muestra la primera variación más notoria en la calidad de resultados como se observa en la figura 4.6. Según se aprecia en la Tabla 4.1, lo más destacable de esta instancia es una flota homogénea de camiones de mediana capacidad, la tasa entre requerimiento y capacidad no es tan elevada por lo que en esta instancia cada configuración tiene un mayor impacto en los resultados. Los tiempos en la instancia 4 siguen mostrando el mismo patrón que las instancias anteriores, se destaca que los valores obtenidos son los más bajos que en cualquier otra instancia, insinuando que la cantidad de reparaciones en esta instancia fue menor que en todas las demás.

La instancia 5 es aquella con una flota con menor libertad de exploración en los algoritmos. Como se ve en la Tabla 4.1 la flota homogénea de camiones posee una de las menores capacidades en cada camión. Esto implicó que todas las configuraciones convergieran rápidamente en términos de calidad. Excepto para la configuración $C5$, donde el armado inicial aleatorio ayuda a una exploración mayor, entregando una gran variedad de soluciones de distintas calidades. Los tiempos no muestran mayor diferencia con los casos anteriores.

La instancia 6 presenta una gran homogeneidad en los resultados tanto respecto a costos como tiempos. Considerando que la instancia 6 posee una alta tasa entre requerimientos y calidad como muestra la tabla 4.1. Además, como se solicita un alto volumen del producto, no es de sorprender que los costos sean tan elevados. Lo que es llamativo es que haya tan poca diferencia entre los resultados obtenidos en cada configuración, considerando las diferencias en el desempeño de las configuraciones en las instancias anteriores. Este resultado podría explicarse si la solución óptima de esta instancia posee un valor cercano al mostrado en esta figura.

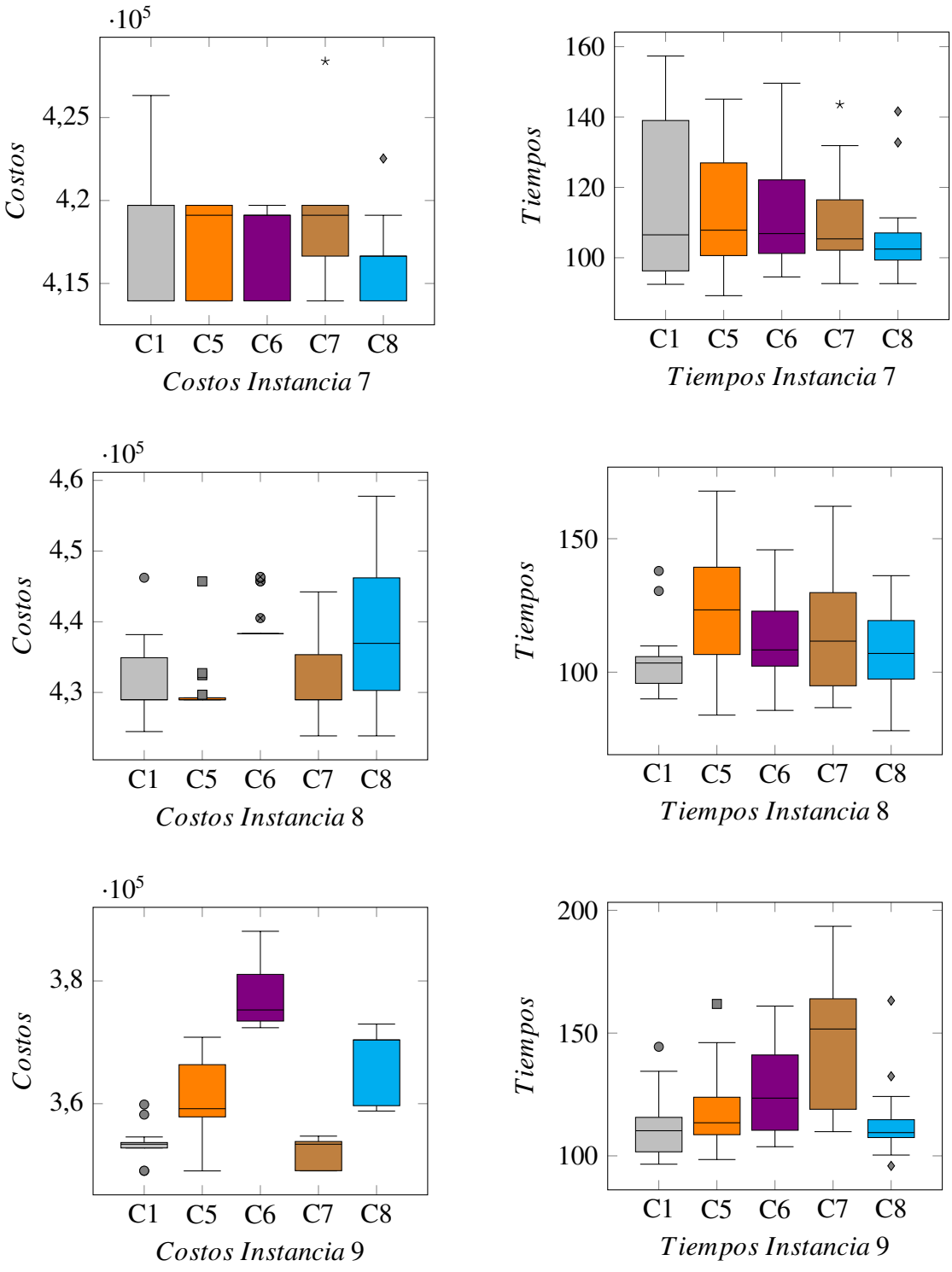


Figura 4.7: Comparación entre los resultados obtenidos para las instancias 7-9 con las configuraciones C1,C5,C6,C7,C8

La instancia 7 tiene una gran similitud con la instancia 6. Ambas instancias poseen casi las mismas características diferenciándose solamente en que un camión grande de la instancia 6 fue divido en dos camiones pequeños como se muestra en la tabla 4.1. Esto se ve reflejado en los resultados que se presentan en la figura 4.7. Los costos aumentaron nuevamente, implicando que el algoritmo encuentra mejores soluciones con flotas pequeñas con alta capacidad de carga, que con grandes flotas de camiones con pequeñas capacidades. Notar que, a pesar de que la figura pareciera mostrar que las configuraciones obtuvieron un mayor rango de soluciones, los índices en el eje *y* son de intervalos más pequeños. Curiosamente los tiempos para la instancia 7 no siguen el patrón que habían seguido en las instancias ya analizadas, si bien los menores tiempos de cada configuración siguen siendo relativamente bajos, no mayores a los 100 segundos, se ven diferentes anomalías. Por ejemplo, la configuración $C1$ no muestre el menor tiempo, y que $C8$ haya sido la que, en promedio, obtuvo menores tiempos. En esta sección, los tiempos están fuertemente relacionados con la cantidad de reparaciones que se deben realizar en cada iteración, una menor demora implica menor cantidad de reparaciones, lo que a su vez implica que la construcción inicial entregó una solución más cercana a la solución final. Notar que todas las configuraciones llegaron a la solución óptima.

Es curioso notar que la instancia 8 muestra peores resultados que la instancia 7, considerando que la instancia 8 posee una mayor capacidad a costa de poseer un camión más en la flota de vehículos. Esto refuerza la idea de que una flota de menor cantidad de vehículos con mayores capacidades entrega mejores resultados. Siendo la instancia 8 la cual posee mayor complejidad. Las configuraciones $C5$ y $C6$ convergieron rápidamente y fueron las únicas que no llegaron a la solución óptima entre las cinco configuraciones comparadas. En el caso de $C6$ el armado greedy basado en producción no pareciera entregar buenos resultados cuando se trabaja con capacidades de carga pequeña. Por su parte, la configuración $C5$, podría no obtener buenos resultados cuando se trata de estructuras más complejas, es decir, una flota más grande.

En la instancia 9 los costos bajaron considerablemente a pesar de ser la instancia que necesitaba una mayor cantidad del producto, pero con una flota de camiones más pequeña. Tanto $C1$ como $C5$ y $C7$ lograron alcanzar la mejor solución. Curiosamente $C6$ no logro obtener buenos resultados a pesar de trabajar con vehículos con una gran capacidad, esto

puede atribuirse a que, llenando los camiones con la máxima capacidad posible, para obtener la cantidad de leche requerida se necesitan visitar varios nodos de una producción menor, aumentando el costo debido a la distancia recorrida.

Vale notar que a pesar de que los costos en la instancia 9 fueron menores que en la instancia 8, los tiempos de la ultima instancia fueron superiores en todas las configuraciones, especialmente en $C7$, probando que el resultado en tiempo de las instancias no tiene una relación directa con la calidad obtenida.

Realizando un análisis general en todas las instancias anteriores (Figuras 4.5, 4.6 y 4.7), se analizaron los gráficos de las configuraciones $C1$, $C5$, $C6$, $C7$, $C8$. Una de las primeras cosas que se puede notar es que los tiempos de ejecución tienden a tener valores cercanos ya que ahora se trabaja con todas las configuraciones con parámetros casi iguales y lo único que varia es el armado inicial. De todas formas sigue existiendo una ventaja comparando los valores de tiempos en la configuración $C1$, lo que resulta más evidente al revisar las medias en cada boxplot de las instancias $1-9$. Esto puede darse por distintos motivos, si bien la cantidad de operaciones al inicio de cada fase de construcción es similar, puede que la construcción por distancia ofrezca una mayor diversidad de nodos al armar los recorridos. Esto puede resultar en una menor cantidad de reparaciones luego de las operaciones de cruzamiento. Esto mismo puede ser el motivo de que los tiempos de $C6$ y $C7$ sean mayores al resto en general. Al moverse directamente a los nodos con mayor cantidad de leche y los que generar menor costo, arman soluciones con poca diversidad. Eso lleva a una mayor cantidad de reparaciones y la necesidad de que la mutación genere una diversidad suficiente para que las operaciones de cruzamiento no generen la necesidad de reparaciones.

Es posible notar que el desempeño de las configuraciones presentadas tiende a ser similar entre ellas. En las instancias 1, 3, 6 y 7 todas las configuraciones estudiadas en esta sección, $C1$, $C5$, $C6$, $C7$, $C8$, encuentran la misma mejor solución. Si bien los valores presentados son similares en dichas instancias cabe analizar que sucede en las esto no se cumple (Instancias 2, 4, 5, 8 y 9). Analizando el cuadro 4.7 es posible notar que las instancias 8 y 9 son aquellas que tienen mayor capacidad de carga sobre la demanda total. En el caso de las instancias 3, 6 y 7, estas poseen una menor carga volviendo a indicar que, teniendo más espacio para acomodar los nodos en el recorrido, la exploración se vuelve mayor, mostrando una mayor

diversidad de soluciones en las instancias, explicando el comportamiento antes nombrado.

En las instancias con mayor diferencia se puede apreciar que las configuraciones de armado greedy basado en producción tienden a obtener los peores resultados en términos de costo. Esto puede deberse a factores, como que haya un gran densidad de nodos cerca de la planta procesadora, logrando mejores resultados con otros armados. También puede deberse a que los nodos con mayor cantidad de leche se encuentren mucho más lejos de la fabrica. También puede deberse a que en algunos casos, ingresar un nodo con mayor producción de leche resulte en que la capacidad restante solo permita ingresar nodos lejanos que posean menor producción.

El armado greedy basado en costos abarca una de las mayores variaciones en costos, esto podría implicar que, ya que los armados se inician tomando camiones aleatorios, las rutas armadas son siempre muy distintas a otras dependiendo de la semilla. Esto se debe a que los primeros camiones que aseguran los nodos de menor costo, obligando al resto de los camiones a visitar una mayor cantidad de nodos. Esto se acentúa si los camiones con mayor capacidad de recolección son los últimos a los cuales se les asignan las granjas en sus recorridos. No es coincidencia que la instancia 5 sea la única donde la mayoría de las versiones de construcción converjan a un único valor. Esto ya que es esta instancia todos los camiones comparten la misma capacidad y sus capacidades son menores, limitando la posibilidad de visitas a granjas de mayor producción. Curiosamente se esperaba que $C2$ y $C8$ fueran las configuraciones con mayor entrega de soluciones, siendo $C8$ implementada con el fin de que hubiera una mayor exploración de soluciones, pero manteniendo una explotación constante al momento de realizar el armado inicial, pero no se contó con que estas configuraciones convergerían

Comparando ambas secciones, es decir, la sección `4.3.1` (Figuras 4.2, 4.3 y 4.4) y `4.3.2` (Figuras 4.5, 4.6 y 4.7), las configuraciones que entregaron mejores resultados fueron $C1$, $C3$, $C5$ y $C7$ y las peores fueron $C2$ y $C4$. Como fue mencionado anteriormente, el método de selección torneo-2 fue perjudicial para la búsqueda de resultados a lo largo del proceso, mientras que en la segunda sección todas las configuraciones tuvieron un buen desempeño, siendo $C6$ la configuración que obtuvo peores resultados pero aun así aceptables. Uno de

los detalles a recalcar es que los mejores resultados obtenidos en la mayoría de las configuraciones tienden a los mismos valores. Los casos más extremos fueron las instancias 1 y 5 donde los boxplots mostraron simplemente una linea. Se teoriza que estas configuraciones convergieron a un resultado en la mayoría de sus soluciones después de una cierta cantidad de iteraciones, ya que nunca se trabajo con soluciones infactibles, no fue posible seguir explorando otras alternativas para optimizar dichos valores.

Se puede concluir analizando todas las figuras que el tiempo no tiene una relación directa con la calidad de las soluciones. En la primera sección las configuraciones $C2$ y $C4$ tenían constantemente tiempos más reducidos siendo que sus costos eran siempre mayores a los de las configuraciones $C1$ y $C3$ ((Figuras 4.2, 4.3 y 4.4). Por otro lado, la segunda sección de experimentos (4.5, 4.6 y 4.7) $C1$ tendía a tener los mejores tiempos y solía obtener las mejores soluciones en la mayoría de los casos.

4.3.3. Comparación con la literatura

Este trabajo se comparara con el los resultados presentados por Montero *et al.* en [9]. El artículo es el único que posee instancias disponibles para comparación. Los autores presentaron un modelo de programación lineal entera (MIP) y un algoritmo Greedy Randomized Adaptative Search Process ($GRASP$) para resolver las instancias estudiadas.

Los resultados obtenidos por los autores se presentan en el cuadro 4.6. En la tabla se presentan los mínimos costos obtenidos en cada caso, así también como los tiempos de ejecución. El GAP (o brecha) respecto a la solución de mejor calidad se utiliza para comparar los resultados de los dos algoritmos comparados. Este mismo valor se utilizo para calcular la diferencia entre los tiempos y valores obtenidos por la configuración $C3$ y los mejores valores del articulo presentado por Montero *et al.*, este valor esta definido como:

$$GAP = \frac{Solucion - VC}{VC} \tag{4.2}$$

	MIP		GRASP			
Inst.	**Min.**	**T[s]**	**Min**	**Prom.**	**GAP %**	**T[s]**
1	111071	3600	115860	117587	4.31	1200
2	191624	3600	197456	201089	3.04	1200
3	359583	3600	370343	393284	2.99	1200
4	214088	3600	218048	222123	1.85	1200
5	242963	3600	243625	243750	0.27	1200
6	435756	3600	388048	398537	-10.95	1200
7	440296	3600	431530	451423	-1.99	1200
8	453439	3600	436864	460880	-3.79	1200

Cuadro 4.6: Resultados obtenidos por Montero *et al.* (2019)

Para realizar la comparación con el estado del arte se utilizo la configuración $C3$, configuración en la cual se utilizo un armado Greedy en base a la menor distancia recorrida, una selección aleatoria y una búsqueda local en la fase de elitismo para mejorar las soluciones (La configuración $C1$, que se referenciara en un momento, tiene los mismo parámetros de configuración excepción de que no utiliza la busqueda local).A pesar de que ParamILS entrego resultados que indicaban que la mejor configuración era $C1$, lo cual es cierto para la mayoría de las soluciones ya que $C1$ entrega mejores soluciones con mayor consistencia, pero en este estudio, considerando outlayers, $C3$ obtiene mejores resultados encontrados en todas las instancias por todas las configuraciones.

	C3			Montero				
Inst.	**Min.**	**Prom.**	**T[s]**	**Min**	**Prom.**	**T[s]**	**GAP % Min**	**GAP % Prom.**
1	118231	118231	87.7	111071	117587	3600	6.45	0.55
2	191624	195227.65	138.21	191624	201089	3600	0.00	-2.91
3	349050	351995.75	103.89	359583	393284	3600	-2.93	-10.50
4	216696	220819.3	86.41	214088	222123	3600	1.22	-0.59
5	242963	243094.2	76.43	242963	243750	3600	0.00	-0.27
6	368295	375435.2	143.57	388048	398537	1200	-5.09	-5.80
7	413953	419958.4	99.66	431530	451423	1200	-4.07	-6.97
8	423855	432424.8	111.75	436864	460880	1200	-2.98	-6.17

Cuadro 4.7: Comparación de resultados obtenidos en $C3$ con los mejores del estado del arte.

En la tabla 4.7 se armo una mezcla de los mejores resultados obtenidos, tanto por el método *GRASP* (aquellos valores cuyo tiempo sea igual a 1200 segundos) como por el método *MIP* (aquellos valores cuyo tiempo sea igual a 3600 segundos). Los resultado de la comparación se listan a continuación.

- La configuración $C3$ de la aproximación de algoritmo genético obtiene la ventaja en cuatro instancias en términos de costos (Instancias 3, 6, 7 y 8). Notar que el modelo de programación linea (*MIP*) esta construido como una técnica de búsqueda de soluciones completa, por ende, para problemas de mayor complejidad es esperable que no obtenga mejores resultados que una búsqueda incompleta (a menos que el algoritmo *MIP* se ejecute durante una inmensa cantidad de tiempo), en comparación del algoritmo *GRASP*, el algoritmo genético se puede basar en este y mejorarlo, por lo cual no es de sorprender que obtenga mejores resultados.

- La configuración *MIP* obtiene la ventaja en dos instancias en términos de costo (Instancias 1 y 4), esto es debido a que *MIP* si llega a un resultado óptimo en la busqueda completa mas la configuración $C3$ no lo logra.

- El resultado de las instancias 2 y 5, en términos de costos, fue un empate entre ambas aproximaciones, para la instancia 2 se asume que ambos algoritmos alcanzaron un valor muy cercano al óptimo ya que ambos algoritmos deberían tener un buen desempeño en esta instancia, esta explicación puede ser utilizada para la instancia 5, esto se refuerza al verificar que toda la población de soluciones en $C3$ convergen cerca del mismo resultado.

- En términos de tiempo la aproximación de algoritmo genético obtuvo mejores resultados, con tiempos menores a $1000[s]$ en cada instancias.

4.3.4. Resumen

En esta sección se presentaron todos los experimentos realizados, todas las configuraciones utilizadas para probar el acercamiento de algoritmo genético y la otras heurísticas usadas, como búsqueda local o $Torneo - 2$. Para que la experimentación entregara los mejores valores posibles, se utilizo el software *ParamILS* para sintonizar el algoritmo y obtener los mejores parámetros posibles, dentro de las variables especificadas para la sintonización. Las pruebas del algoritmo fueron bastante satisfactorias, obteniendo excelentes valores en el tiempo de ejecución y obteniendo valores de costo que competerían con el estado del arte comparado. El algoritmo genético entrego los mejores resultados posibles (en términos de costo) cuando las instancias trabajadas utilizaban flotas más pequeñas pero con camiones con altas capacidad de recolección, esto les da libertad a los camiones para elegir cualquier nodo sin preocuparse de superar el limite máximo de capacidad, y además pueden realizar más viajes antes de volver a la fabrica procesadora. De todas las configuraciones utilizadas, las más destacadas fueron, $C1$, una configuración del algoritmo genético estándar con un armado inicial basado en algoritmo *Greedy* prefiriendo las distancias recorridas más pequeñas, esta configuración fue la elegida por el software *ParamILS* siendo la que entrego resultados de mejor calidad de manera más confiable. La segunda configuración fue $C3$, la cual, al igual que $C1$ es una configuración del algoritmo genético estándar con un armado inicial basado en en algoritmo *Greedy* prefiriendo las distancias recorridas más pequeñas, pero, posee una variación en la fase de elitismo, en la cual se realiza una operación de búsqueda local para

disminuir la distancia recorrida del camión ordenando los nodos (granjas) visitados en los recorridos. Si bien esta configuración entrego buenos resultados, no fueron tan confiables como los de $C1$. Ambas configuraciones tuvieron casi los mismo mejores resultados pero $C3$ obtuvo una ventaja sobre $C1$ en la octava instancia probada en esta sección, transformando a $C3$ en la configuración con mayor cantidad de mejores resultados, comparativamente a las otras configuraciones presentadas en este estudio.

Conclusiones

En este articulo se propuso una forma de resolver el problema *PriceCollecting* aplicado a la recolección de leche en el sur de Chile. Se desarrollo un algoritmo genético implementando distintas configuraciones las cuales combinaba distintas heurísticas para resolver el problema, todas las instancias evaluadas fueron sacadas de un problema real. Con los resultados obtenidos, utilizando un algoritmo genético, se puede concluir que las mejores soluciones son obtenidas cuando se utilizan menos camiones con mayores capacidades, esto se debe a que tener camiones con menor capacidad de recolección limita las posibilidades al momento de elegir la siguente granja que sera agregada al recorrido, obligando a agregar granjas con pequeñas producciones aumentando el recorrido del camión, se teoriza que si se trabaja con soluciones infactibles en la búsqueda de soluciones, tener poca capacidad extra sobre la capacidad demandada no debería suponer un problema para obtener buenos resultados.

Los resultados obtenidos fueron satisfactorios, pero se cree que es posible obtener aun mejores, manteniendo el algoritmo genético como enfoque para resolver el problema en cuestión. Como trabajo futuro seria posible realizar dos fases en la búsqueda de soluciones, la primera constaría de una fuerte exploración con posibles soluciones infactibles, para que en la segunda fase se realiza una explotación para converger en las mejores soluciones factibles del problema, estas fases podrían alternarse para tener un balance entre la explotación y la exploración, de esta manera herramientas que parecieron no entregar buenos resultados (como la selección de torneo-2) puedan ser utilizadas en la segunda fase pudiendo influenciar positivamente en las soluciones finales. Sobre los distintos armados iniciales, algunas no mostraron resultados variados y entraron en la explotación demasiado rápido, como $C6$, mas otras herramientas podrían ser reformadas para encontrar obtener mejores valores ($C8$ elije

exclusivamente el nodo con mayor producción luego de realizar una preselección en base a distancia, en un método enfocado a la exploración, el nodo elegido debería ser distinto al de mayor producción). Si bien la búsqueda local fue efectiva para mejorar soluciones, en instancias donde los recorridos sean mucho más largos su efectividad en encontrar mejores soluciones seria mucho mayor en promedio, de cualquier manera gracias a esta heurística se obtuvieron las mejores soluciones.

Bibliografía

[1] Kris Braekers, Katrien Ramaekers, and Inneke Van Nieuwenhuyse. The vehicle routing problem: State of the art classification and review. *Computers Industrial Engineering*, 99:300 – 313, 2016.

[2] Massimiliano Caramia and Francesca Guerriero. A milk collection problem with incompatibility constraints. *INFORMS Journal on Applied Analytics*, 40(2):130–143, 2010.

[3] Maria Caria, Giuseppe Todde, and Antonio Pazzona. Modelling the collection and delivery of sheep milk: A tool to optimise the logistics costs of cheese factories. *Agriculture*, 8(1), 2018.

[4] Mohamed Cheikh, Mustapha Ratli, Omar Mkaouar, and Bassem Jarboui. A variable neighborhood search algorithm for the vehicle routing problem with multiple trips. *Electronic Notes in Discrete Mathematics*, 47:277 – 284, 2015. The 3rd International Conference on Variable Neighborhood Search (VNS'14).

[5] G.D.H. Claassen and Th.H.B. Hendriks. An application of special ordered sets to a periodic milk collection problem. *European Journal of Operational Research*, 180(2):754 – 769, 2007.

[6] Jean-François Cordeau, Gilbert Laporte, and Anne Mercier. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational research society*, 52(8):928–936, 2001.

[7] Mohamed Haouari and Jouhaina Chaouachi Siala. A hybrid lagrangian genetic algorithm for the prize collecting steiner tree problem. *Computers Operations Research*, 33(5):1274 – 1288, 2006.

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

[9] Elizabeth Montero, Darío Canales, Germán Paredes-Belmar, and Raúl Soto. A prize collecting problem applied to a real milk collection problem in chile. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2019)*, pages 1416–1423, Wellington, New Zealand, June 2019.

[10] Germán Paredes-Belmar, Armin Lüer-Villagra, Vladimir Marianov, Cristián E. Cortés, and Andrés Bronfman. The milk collection problem with blending and collection points. *Computers and Electronics in Agriculture*, 134:109 – 123, 2017.

[11] Odivaney Pedro, Rodney Saldanha, and Ricardo Camargo. A tabu search approach for the prize collecting traveling salesman problem. *Electronic Notes in Discrete Mathematics*, 41:261 – 268, 2013.

[12] F. Rojas and F. Meza. A parallel distributed genetic algorithm for the prize collecting steiner tree problem. In *2015 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 643–646, Dec 2015.

[13] Niaz A Wassan and Gábor Nagy. Vehicle routing problem with deliveries and pickups: modelling issues and meta-heuristics solution approaches. *International Journal of Transportation*, 2(1):95–110, 2014.