

# INF-295: Inteligencia Artificial

## Informe final: Milk Collection with Blending

Anghelo Carvajal  
201473062-4

11 de enero de 2021

### Evaluación

Mejoras 1ra Entrega (10 %):	_____
Código Fuente (10 %):	_____
Representación (15 %):	_____
Descripción del algoritmo (20 %):	_____
Experimentos (10 %):	_____
Resultados (10 %):	_____
Conclusiones (20 %):	_____
Bibliografía (5 %):	_____
<b>Nota Final (100):</b>	_____

## Resumen

Este trabajo investigativo presenta el actual estado del arte del problema *Milk Collection with Blending*, el cual es una sub-variante de *Vehicle Routing Problem* (VRP). Este problema busca recolectar leche sin procesar de un conjunto de granjas, donde la mayor diferencia con VRP es que los distintos tipos de leche sin procesar pueden mezclarse entre sí, lo cual produce una leche de calidad inferior. Se definirá el problema, mostrando variables y restricciones típicas, y se mostrará un modelo matemático que modela el problema de forma certera.

## 1. Introducción

El presente documento tiene como propósito presentar y definir el problema *Milk Collection with Blending*, además de documentar el actual estado del arte de dicho problema.

Se comenzará definiendo el problema actual, hablando de forma general de las variables y restricciones típicas de este problema. Luego se expondrá el actual estado del arte del problema *Milk Collection with Blending*, documentando sus orígenes (de que tipo de problema proviene), como han sido enfrentados dichos orígenes, los algoritmos que intentan resolverlo, y las variantes de este mismo problema.

También se presentará un modelo matemático de programación lineal que puede representar este problema de forma certera. Finalmente se expondrán las conclusiones de este trabajo investigativo.

## 2. Definición del Problema

El problema *Milk Collection with Blending* es un problema NP-duro, el cual surge en el año 2016 [8].

Este problema consiste en encontrar un conjunto de rutas óptimas para el recorrido de cada uno de los camiones que se tienen a disposición. Cada una de estas rutas debe proveer un camino para cada camión, de modo que este recoja toda la leche de cada una de las granjas que se le asigne y la lleve a la planta procesadora.

Cada ruta puede empezar en cualquier parte, pero siempre debe terminar en la planta procesadora. Hay un costo asociado al desplazamiento de un camión entre una granja y la otra.

La leche se categoriza en distintos tipos según su calidad. Estos tipos son ordenables de mejor a peor calidad, y las ganancias monetarias también son distintas según dicha calidad. La planta procesadora exige una cantidad mínima de cada tipo (calidad) de leche.

Los camiones pueden transportar una cantidad limitada y no tienen compartimientos separados para cada tipo de leche, por lo que si un camión recoge leches de distintas calidades de las granjas, estas se mezclan dentro del camión, resultando en leche que se considera de la peor calidad de la mezcla. La ventaja de esto es reducir el costo de movilización de los camiones a cambio de menores ganancias por la calidad de la leche.

Los parámetros del problema son los caminos entre las granjas de leche (representadas por un grafo),

los caminos entre las granjas y la planta, y el costo de desplazar a un camión a través de cada uno de estos arcos. La cantidad de camiones de las que se dispone y la capacidad que cada uno puede transportar, las clasificaciones para los tipos de leche, que tipo y cuanta cantidad de leche produce cada granja. Cuanta cantidad de leche de cada tipo exige la planta procesadora en total. Granja en que cada camión inicia su ruta.

Las variable principal de este problema es el orden en el cual cada camión recorre sus granjas, lo cual implica que tipos de leche recogería dicho camión de cada granja, que tipo de leche resultante entrega el camión a la planta y en que cantidad.

Este problema se ve restringido por la capacidad máxima que tiene cada camión, que cada camión recolecte toda la leche de la granja a la que va a recolectar, que cada granja sea visitada por a lo más un camión, cada camión tiene a lo más 1 ruta, se debe respetar la cantidad de leche de cada tipo que exige la planta como mínimo.

El objetivo de este problema es maximizar los beneficios monetarios, disminuyendo los recorridos de los camiones y aumentando la ganancia producida por la leche recolectada según su tipo.

También existen otras variantes de este problema, como que cada camión pueda poseer distintos compartimientos para transportar la leche [4], de modo que 1.- no habría mezcla de leches o 2.- que se minimice la mezcla de tipos de leche; que existan puntos de recolección a los cuales las granjas acercan la leche [7]; o que algunas granjas no sean accesibles por grandes camiones [1].

## 3. Estado del Arte

Este problema es una variación del problema *Vehicle Routing Problem* (VRP), el cual fue documentado por primera vez en el año 1959 [2]. En dicho problema se discutía el encontrar un conjunto de rutas óptimas para una determinada cantidad de vehículos los cuales deben entregar paquetes a los respectivos clientes.

Específicamente este problema no ha sido tan trabajado e investigado debido a ser un problema no tan antiguo. Algunos de los problemas similares y métodos confeccionados para resolver esos y este problema son:

- El primer VRP fue realizado y documentado por Dantzig y Ramser en 1959 [2]. Este se basaba en un problema de distribución de combustible, y

apuntaba a repartir paquetes a clientes geográficamente separados usando un conjunto de vehículos. Aquí se concluye que este tipo de problema es NP-completo, debido a que podía ser reducido a un problema de vendedor viajero.

- A la fecha, la implementación que mejores resultados ha dado es la de Taillard [11], la cual se basa en algoritmos de búsqueda tabú, en donde distribuye a los nodos en dos formas distintas, uno de forma uniforme y la otra de forma no euclidiana.
- Otro tipo de problema basado en VRP es la variante con múltiples productos (MPVRP), el cual ha sido enfrentado con algoritmos genéticos [3], el algoritmo de Dijkstra [6] y modelos de programación lineal entera en conjunto a un solver [5] por nombrar algunos.
- Una variante similar a MPVRP es el problema de enrutamiento de vehículos con múltiples compartimientos (MCVRP). Un método notable es el de El Fallahi et. al. [4], dado que lo resuelven de tres formas distintas; con una heurística constructiva sin iteraciones de mejora, con búsqueda tabú y con un algoritmo memético (el cual es a su vez una extensión de los algoritmos genéticos).
- La primera vez que se atacó de la recolección de leche como una variante específica de VRP fue en el año 1994 por Sankaran y Ubgade [9] para resolver un caso real de 70 granjas en India. Ellos consideraron camiones de diferentes capacidades y no poder exceder ciertos límites de tiempo entre cada recolección dado a las distintas condiciones climáticas de la zona.
- El problema de la recolección de leche que admite la mezcla de los distintos tipos de leche (nuestro problema) fue abordado por primera vez el año 2016 por Germán Paredes-Belmar et. al [8]. En dicha ocasión se formuló el problema a través de un modelo entero mixto. Para resolver instancias medianas usaba un algoritmo de bifurcación y corte. Para instancias más grandes se utilizaba un procedimiento heurístico, el cual consiste en dividir la instancia real en conjuntos de granjas, las cuales eran repartidas según su ubicación. En este estudio se discute el mezclar los tipos de leches vs no mezclarlos al transportarlos en los camiones, y concluía que mezclar las leches predominaba por sobre no mezclarlos, debido a la viabilidad y relajación que esta otorga. También estudia como afecta los camiones con múltiples compartimientos y con compartimiento único a esta variante.
- En el año 2017, los mismos autores enfrentan una variante del problema [7]. En este problema se tenía una cantidad aún mayor de granjas, por lo que el modelo anterior no podía encontrar una

solución en tiempo razonable, por lo que agregaron puntos de recolección para cada conjunto de granjas, lo cual permitió encontrar soluciones en tiempos mucho más acotados.

- Villagran propone 2 acercamientos, basados técnicas de búsqueda local, para este problema, en el año 2019 [12]. Estos algoritmos eran basados en las técnicas *hill-climbing* e *iterated local search*, las cuales ambas entregaron soluciones de buena calidad, de modo que abre la puerta a la posibilidad de no usar técnicas completas para la resolución de este problema. En las pruebas del autor, estos algoritmos lograron obtener el óptimo global en la mayoría de instancias utilizadas. Cabe destacar que ambas implementaciones entregaban mejores resultados, tanto en calidad de la solución como en tiempo de ejecución, que los algoritmos del estado del arte hasta esa fecha.
- En el año 2019, Soto [10] propone un acercamiento basado en la meta-heurística *Simulated Annealing*, el cual tiene un muy buen desempeño en instancias pequeñas y medianas, comparables al solver CPLEX.

## 4. Modelo Matemático

A continuación se presenta un modelo de programación entera mixta, tal como es descrito en [12], el cual está basado en [8].

### 4.1. Función objetivo

Este problema tiene como objetivo la maximización del beneficio monetario, por lo que se considera la diferencia entre la ganancia producida por la leche recolectada vs los costos de transporte de las rutas construidas.

Esto se representa a través de:

$$Max \left\{ \sum_{t \in T} \sum_{r \in R} \alpha^r v^{tr} - \sum_{(i,j,k \in AK)} c_{ij}^k x_{ij}^k \right\} \quad (1)$$

### 4.2. Parámetros

Los parámetros de este problema son los siguientes:

- $A$ : Conjunto de arcos que representan caminos entre productores de leche.
- $A^0$ : Conjunto de arcos que representan caminos entre planta y productores de leche.
- $N$ : Conjunto de productores,  $N = 0, \dots, n$ . Se consideran en total  $n$  productores.
- $N_0$ : Conjunto de productores y la planta.
- $K$ : Conjunto de camiones
- $T$ : Conjunto de calidades de leche

- $N^t$  : Conjunto de productores de leche de calidad  $t \in T$ .
- $D^t$  : Resultado de la mezcla de leche de calidad  $r$  con leche de calidad  $t$ .
- $IT$  : Conjunto de pares ordenados  $(i, t)$  de productores  $i$  y leche de calidad  $t$ , donde cada cliente produce solo una calidad de leche.
- $Q^k$  : Capacidad de cada camión  $k$ .
- $q_i^t$  : Cantidad de leche  $t$  producida por el productor  $i$ .
- $c_{ij}^k$  : Costo del viaje de cada camión  $k$  sobre el arco  $(i, j) \in A \cup A^0$ .
- $\alpha^t$  : Ingreso por unidad leche de calidad  $t$ .
- $P^t$  : Requerimientos de leche de calidad  $t$  de la planta.

### 4.3. Variables de decisión

Este modelo posee 5 variables de decisión. De estas, 3 son binarias:

- $x_{ij}^k$ , la cual vale 1 si el camión  $k$  viaja directamente del nodo  $i$  al nodo  $j$ . Espacio de búsqueda:  $2^{|N|*|N|*|K|}$
- $y_i^{kt}$ , la cual vale 1 si el camión  $k$  recoge leche de calidad  $t$  de la granja  $i$ . Espacio de búsqueda:  $2^{|N|*|K|*|T|}$
- $z^{kt}$ , la cual vale 1 si el camión  $k$  entrega leche de calidad  $t$  a la planta. Espacio de búsqueda:  $2^{|K|*|T|}$

Las otras 2 variables no binarias:

- $w^{kt}$ , la cual indica el volumen de leche de calidad  $t$  que el camión  $k$  entrega a la planta. Espacio de búsqueda:  $|Q|^{|K|*|T|}$
- $v^{tr}$ , la cual indica el volumen de leche de calidad  $t$  entregada a la planta, mezclada para su uso como leche de calidad  $r$ . Espacio de búsqueda:  $|Q|^{|T|*|T|}$

#### 4.3.1. Espacio de búsqueda

El espacio de búsqueda es:

$$2^{|N|*|N|*|K|} * 2^{|N|*|K|*|T|} * 2^{|K|*|T|} * |Q|^{|K|*|T|} * |Q|^{|T|*|T|} \quad (2)$$

Lo cual puede ser reescrito como:

$$2^{|K|*(|N|^2+|N|*|T|+|T|)} * |Q|^{|T|*(|K|*|T|)} \quad (3)$$

### 4.4. Restricciones

Este modelo se ve restringido por las siguientes restricciones:

La restricción 4 limita la cantidad de leche que puede recolectar cada camión de acuerdo a su capacidad. Se considera una flota heterogénea.

$$\sum_{r \in T} \sum_{i \in N: (i, j) \in IT} q_i^t y_i^{kt} \leq Q^k, \forall k \in K \quad (4)$$

La restricción 5 establece que la recolección de la leche de cada productor debe ser realizada por exactamente un camión. Esto implica que se debe recolectar la leche de todos los productores y que un productor no puede ser visitado más de una vez.

$$\sum_{k \in K_i} y_i^{kt} = 1, \forall i \in N, t \in T : (i, j) \in IT \quad (5)$$

La restricción 6 establece que cada camión debe tener como máximo una ruta la cual comienza desde la planta.

$$\sum_{j: (0_k, j, k) \in AK} x_{0_k j}^k \leq 1, \forall k \in K \quad (6)$$

La restricción 7 permite controlar el flujo para el orden de las visitas de los nodos por parte de cada camión.

$$\sum_{i: (i, j, k) \in AK} x_{ij}^k = \sum_{h: (j, h, k) \in AK} x_{jh}^k, \forall k \in K, j \in N_0 \quad (7)$$

La restricción 8 establece que cada camión que visita cada granja debe detenerse y recoger su leche.

$$\sum_{p: (p, i, k) \in AK} x_{pi}^k = y_i^{kt}, \forall k \in K, i \in N, t \in T : (i, t) \in IT \quad (8)$$

Las restricciones 9, 10, 11 y 12 establecen las reglas de mezcla de leche. La restricción 9 controla el tipo de leche de cada camión de acuerdo a cada una de las granjas que ha visitado. La restricción 10 controla que cada camión entrega en planta solo un tipo de leche. La restricción 11 mide la cantidad de leche entregada de cada tipo de acuerdo a la capacidad de los camiones que recolectaron cada tipo de leche. Por último, la restricción 12 mide la cantidad efectivamente recolectada de cada tipo de leche considerando las granjas visitadas por cada camión.

$$z^{kt} \leq 1 - \sum_{r \in D^t: r \neq t, (i, r) \in IT} y_i^{kr}, \forall k \in K, i \in N, t \in T \quad (9)$$

$$\sum_{t \in T} z^{kt} \leq 1, \forall k \in K \quad (10)$$

$$w^{kt} \leq z^{kt} Q^k, \forall k \in K, t \in T \quad (11)$$

$$w^{kt} \leq \sum_{r: t \in D^r} \sum_{h \in N^r} q_h^r y_h^{kr}, \forall k \in K, t \in T \quad (12)$$

La restricción 13 se encarga de forzar que cada camión se lleve toda la leche producida por cada granja a la planta.

$$\sum_{k \in K} \sum_{t \in T} w^{kt} = \sum_{(i,t) \in IT} q_i^t \quad (13)$$

La restricción 14 se encarga de equilibrar la cantidad de leche de cada calidad que llega a la planta y la cantidad de leche de cada calidad restante después de la mezcla en la planta.

$$\sum_{r \in D^t} v^{tr} = \sum_{k \in K} w^{kt}, \forall t \in T \quad (14)$$

La restricción 15 se encarga de controlar la satisfacción de las cuotas de leche de cada tipo.

$$\sum_{t \in T} v^{tr} \geq P^r, \forall r \in D^t \quad (15)$$

La restricción 16 evita mezclas de leche prohibidas.

$$y_i^{kt} + y_i^{kr} \leq 1, \forall (t, r) \in PM; (i, t), (j, t) \in IT \quad (16)$$

La restricción 17 evita la aparición de sub-ciclos en las rutas de cada camión.

$$\sum_{i \in S} \sum_{j \in S} x_{ij}^k \leq |S| - 1, \forall S \subseteq N, k \in K \quad (17)$$

Las restricciones 18, 19 y 20 controlan la naturaleza de las variables de decisión del modelo planteado. La restricción 18 controla la naturaleza binaria de las variables asociadas a los tipos de leche recolectados y mezclados en ruta. La restricción 19 controla la naturaleza de la variable binaria que controla las secuencias de visitas de los camiones. La restricción 20 controla la naturaleza no negativa de las variables de volúmenes de leche entregados y mezclados en planta.

$$y_i^{kt}, z^{kt} \in \{0, 1\}, \forall i \in N, k \in K, t \in T : (i, t) \in IT \quad (18)$$

$$x_{ij}^k \in \{0, 1\}, \forall (i, j, k) \in AK \quad (19)$$

$$w^{kt}, v^{tr} \geq 0, \forall k \in K; t, r \in T, r \in D^t \quad (20)$$

Todas estas restricciones se trabajan como restricciones duras, es decir, toda solución al problema debe cumplir con todas las restricciones impuestas.

## 5. Representación

Como este problema tiene como objetivo encontrar rutas óptimas para una determinada cantidad de camiones, donde cada ruta consiste en el conjunto ordenado de granjas que recorrería el camión. Sabiendo esto, una buena representación matemática consiste en un “arreglo de arreglos” o “vector de vectores” (no confundir con matriz).

Cada uno de estos vectores interiores modelarían las rutas que se le asignarían a los camiones, de modo que cada uno de estos vectores interiores contendrían los identificadores de las granjas a recorrer que se están asignando a esta ruta, y en el orden en el que se deben recorrer. De esta forma, el vector exterior contendría el listado de rutas que se le asignarían a cada uno de los camiones.

Se usa esta representación debido a que se adapta bien al problema, ya que:

- Permite tener rutas de distintos con distintas cantidades de granjas a recorrer (a diferencia de una matriz de tamaño fijo).
- Como existen tantas rutas como camiones y cada camión está asignado a una ruta distinta, se maneja en parte la restricción 6.
- Al indicar simplemente que granjas debe recorrer en cada ruta y no almacenar más información, se fuerza a que el camión deba recoger toda la leche de cada granja que recorre, manejando la restricción 8 y 13 de forma implícita.
- Al ser vectores que contienen granjas que no se repiten dentro del mismo vector u en otras rutas, se imposibilita que se generen sub-ciclos, por ende manejando la restricción 17.

A nivel de implementación, se utilizaron clases que ayudan a cuidar la lógica interna y el estado de las variables. La solución final se ve representada por la clase `Solution`, la cual maneja las rutas usando un `std::vector<Route>`.

La clase `Route` modela una ruta asignada a un camión y que transporta un tipo determinado de calidad de leche, además de manejar las granjas a recorrer por dicho camión usando un `std::vector<Node*>`. Se prefirió usar punteros a `Node` en lugar de usar la clase directamente para evitar la reinstanciación al mover una granja de una ruta a otra.

La clase `Node` simplemente contiene los datos de la granja que representa, tal como el identificador, la posición, la cantidad de leche que produce, entre otros.

Si se ignora la abstracción de las clases antes mencionadas, esta implementación se puede interpretar como `std::vector< std::vector<int> >`, lo cual es

muy similar a la representación matemática antes expuesta.

Cabe destacar que una `Route` no incluye la planta procesadora, ya que un camión siempre debe comenzar y terminar su recorrido en ella, por lo que almace-

narla de forma explícita sería un desperdicio de memoria y complicaría los movimientos a realizar. Además, al modelarlo de esta manera, se maneja en parte la restricción 6.

## 6. Descripción del algoritmo

### 6.1. Generación de solución inicial

```

1 function initialSolution(instance):
2   solution ← vector vacío de rutas
3
4   for milk in instance.milktypes in random order do
5     route ← ruta vacía
6     assign milk to route
7     add route to solution
8   end for
9
10  valid ← false
11  while not valid do
12    valid ← true
13    for index in solution in random index order do
14      route ← solution[index]
15      truck ← instance.trucks[index]
16      if truck.capacity < route.quota do
17        valid ← false
18        break
19      end if
20
21      assign truck to route
22    end for
23  end while
24
25  TOL ← 0
26  truck_counter ← 0
27
28  farms ← instance.farms
29  while farms.length > 0 do
30    selected_farm ← select random from farms
31    for route in solution do
32      if route.capacity_left - selected_farm < 0 do
33        continue
34      end if
35      if route.milk_type != selected_farm.milk_type do
36        continue
37      end if
38
39      remove selected_farm from farms
40      add selected_farm to route
41      truck_counter += 1
42      selected_farm ← select random from farms
43    end for
44
45    if truck_counter == 0 do
46      TOL -= 10
47    else
48      truck_counter ← 0
49    end if
50  end while
51
52  return solution
53 end function

```

## 6.2. Función de evaluación

```

1 function evaluateSolution(solution, instance):
2   result ← 0
3   for route in solution do
4     result += evaluateRoute(route, instance.initialNode)
5   end for
6   return result
7 end function

```

```

1 function evaluateRoute(route, initial_node):
2   milk_total ← 0
3   total_distance ← 0
4
5   previous_farm ← initial_node
6   for farm in route do
7     milk_total += farm.produced
8     total_distance += distance(previous_farm, farm)
9     previous_farm ← farm
10  end for
11  total_distance += distance(previous_farm, initial_node)
12
13  quality ← milk_total * milk_profit_percentage
14  quality -= total_distance
15  if milk_total > capacity do
16    quality -= milk_total - capacity
17  end if
18  if milk_total < quota do
19    quality -= quota - milk_total
20  end if
21
22  return quality
23 end function

```

## 6.3. Hill climbing

```

1 function hillClimbing(initial_solution, instance, K):
2   solution ← initial_solution
3   movements ← lista de movimientos
4
5   i ← 0
6   is_better_solution ← true
7   while i < K && is_better_solution do
8     is_better_solution ← false
9     current_quality = evaluateSolution(solution, instance);
10
11     for movement in movements in random order do
12       is_better_solution ← movement(solution, instance, current_quality)
13
14       i += 1
15       if is_better_solution do
16         break
17       end if
18     end for
19   end while
20
21   return solution
22 end function

```

### 6.3.1. Move node between routes

```

1 function moveNodeBetweenRoutes(solution, instance, old_quality):

```

```

2  was_feasible ← isFeasible(solution)
3  for src_route in solution in random order do
4      for dst_route in solution in random order do
5          if src_route == dst_route do
6              continue
7          end if
8
9          for src_farm in src_route in random order do
10             if dst_route.capacity_left < src_farm.produced do
11                 continue
12             end if
13
14             if not canRemoveFarm(src_route, src_farm) do
15                 continue
16             end if
17
18             remove src_farm from src_route
19
20             for dst_farm in dst_route in random order do
21                 if not canAddFarm(dst_route, src_farm) do
22                     continue
23                 end if
24
25                 add src_farm to dst_route
26
27                 if not was_feasible do
28                     capacities_improved ← didCapacitiesLeftImproved(solution)
29                     quotas_improved ← didQuotasDiffImproved(solution)
30                     if capacities_improved || quotas_improved do
31                         return true
32                     end if
33                 end if
34
35                 new_quality ← evaluateSolution(solution, instance)
36                 if new_quality > old_quality do
37                     return true
38                 end if
39
40                 remove src_farm from dst_route
41             end for
42
43             add src_farm to src_route
44         end for
45     end for
46 end for
47
48 return false
49 end function

```

### 6.3.2. 2opt intra-route

```

1 function intraRoute2Opt(solution, instance, old_quality):
2     for route in solution in random order do
3         for left_farm_index in route in random index order do
4             for right_farm_index in route[left_farm+1:] in random order do
5                 reverseOrder(route, left_farm_index, right_farm_index)
6
7                 new_quality ← evaluateSolution(solution, instance)
8                 if new_quality > old_quality do
9                     return true
10                end if
11
12                reverseOrder(route, left_farm_index, right_farm_index)
13            end for

```



```

14     end for
15   end for
16
17   return false
18 end function

```

### 6.3.3. Remove node

```

1 function removeNode(solution, instance, old_quality):
2   was_feasible ← isFeasible(solution)
3   for route in solution in random order do
4     for farm in route in random order do
5       if not canRemoveFarm(route, farm) do
6         continue
7       end if
8
9       remove farm from route
10
11     if not was_feasible do
12       capacities_improved ← didCapacitiesLeftImproved(solution)
13       quotas_improved ← didQuotasDiffImproved(solution)
14       if capacities_improved || quotas_improved do
15         return true
16       end if
17     end if
18
19     new_quality ← evaluateSolution(solution, instance)
20     if new_quality > old_quality do
21       return true
22     end if
23
24     add farm to route
25   end for
26 end for
27
28   return false
29 end function

```

### 6.3.4. Interchange node between routes

```

1 function interchangeNodeBetweenRoutes(solution, instance, old_quality):
2   was_feasible ← isFeasible(solution)
3   for src_route_index in solution in random index order do
4     src_route ← solution[src_route_index]
5
6     for src_farm_index in src_route in random index order do
7       src_farm ← src_route[src_farm_index]
8
9       for dst_route in solution[src_route_index+1:] in random order do
10        if not isMilkTypeCompatible(dst_route, src_farm) do
11          continue
12        end if
13
14        for dst_farm_index in dst_route in random index order do
15          dst_farm ← dst_route[dst_farm_index]
16          if not isMilkTypeCompatible(src_route, dst_farm) do
17            continue
18          end if
19
20          src_route[src_farm_index] ← dst_farm
21          dst_route[dst_farm_index] ← src_farm
22
23          if not was_feasible do

```

```

24         capacities_improved ← didCapacitiesLeftImproved(solution)
25         quotas_improved ← didQuotasDiffImproved(solution)
26         if capacities_improved || quotas_improved do
27             return true
28         end if
29     end if
30
31     new_quality ← evaluateSolution(solution, instance)
32     if new_quality > old_quality do
33         return true
34     end if
35
36     src_route[src_farm_index] ← src_farm
37     dst_route[dst_farm_index] ← dst_farm
38 end for
39 end for
40 end for
41 end for
42
43 return false
44 end function

```

## 7. Experimentos

El algoritmo fue probado con 12 instancias<sup>1</sup> de distintos tamaños. Además, se le agregó colores al output final del programa para poder saber más fácilmente si el resultado final es factible o no y para saber que calidad de leche produce cada granja asignada a cada ruta.

Si la solución final es infactible, se muestra en rojo el beneficio total de esta solución y la letra de la ruta infactible. Si la solución no es factible debido a que una ruta no cumple con la cuota mínima de ese tipo de leche, se muestra en rojo el número que indica la cantidad total de leche que se está llevando de este tipo de esa ruta. Además, se le asignó un color a las granjas dependiendo del tipo de leche que producen; si la granja es de tipo 'A' se mostrará en color verde; si es de tipo 'B' se mostrará en color amarillo y así sucesivamente.

Esta metodología permitió detectar fácilmente que se generaban soluciones infactibles, analizar los datos y confeccionar nuevos movimientos y modificar los ya existentes para mejorar la factibilidad de las soluciones.

El entorno de experimentación consiste en un equipo con procesador Intel Core i5-6400 2.70GHz, con 24GB de memoria RAM a 2133MHz y un disco de duro de 2TB y 5400rpm. El equipo cuenta con Pop!\_ OS 20.10.

El algoritmo requiere que se le entregue como parámetro los datos de la instancia y el máximo de iteraciones a realizar (llámese  $K$ ).

Para poder ejecutar el programa, se le debe entregar como parámetro la ruta a un archivo que contiene los datos de la instancia y el número entero  $K$ .

Este archivo indica la siguiente información:

- La cantidad de camiones y la capacidad de cada uno.

- La cantidad de tipos de leche, la cuota mínima que se requiere de cada tipo y la ganancia que aporta cada uno.
- La cantidad de granjas y la información asociada a cada uno. Se indica un identificador numérico, posiciones x e y, una letra indicando la calidad de esta y la cantidad de leche producida. Cabe destacar que el primero de esta lista no es una granja, si no que se refiere a la planta procesadora.

El programa usa los datos de este archivo como parámetro para el algoritmo.

Se utilizaron 12 instancias distintas, las cuales todas tenían 3 camiones y 3 tipos de leche en cada instancia (variando la cuota pedida y la capacidad de los camiones). La diferencia principal de las instancias son en la cantidad de nodos, variando desde 22 hasta 80 nodos, donde las instancias con mayor cantidad de nodos requerían una mayor cantidad de iteraciones para terminar.

El algoritmo tiene 2 criterios de término. El primero es si la cantidad de iteraciones sobrepasa al parámetro  $K$ , entendiéndose una iteración como la realización de alguno de los movimientos propuestos en la sección anterior. El segundo criterio de parada consiste en la detección de un óptimo local, de modo que ninguno de los movimientos propuestos puede mejorar la calidad de la solución.

Todas las instancias fueron probadas repetidas veces con un  $K = 1000$ , pero ninguna ejecución sobrepasó las 500 iteraciones. Normalmente oscilaban entre las 60 (instancias pequeñas) y las 460 iteraciones (instancias más grandes).

<sup>1</sup>[https://github.com/AngheloAlf/2020-2\\_IA\\_Milk\\_collection\\_problem\\_with\\_blending/tree/master/instances](https://github.com/AngheloAlf/2020-2_IA_Milk_collection_problem_with_blending/tree/master/instances)

## 8. Resultados

Que fue lo que se logró con la experimentación, incluir tablas y parámetros, gráficos (por ejm boxplot), lo más explicativo posible. En los resultados se espera que concluya cuál fue el rendimiento del algoritmo con los experimentos detallados en la sección anterior, y compare las diferencias entre configuraciones distintas de los experimentos. Analizar los resultados obtenidos y concluir acerca de aspectos del algoritmo y/o de la complejidad de las instancias, o acerca de características relacionadas con su implementación.

## 9. Conclusiones

Como cualquier otro problema NP-duro, este problema CSOP ha demostrado ser difícil de modelar de modo que entregue una solución óptima en un tiempo prudente.

Debido a que es un problema bastante joven (aunque VRP data de 1959 [2], esta variante específica se puede encontrar en la literatura desde el año 2016 [8]) se han desarrollado escasos modelos y métodos de resolución para este problema específico.

De todas formas, las técnicas que se han usado para trabajar *Milk Collection with Blending* suelen atacar al mismo problema base, pero con leves diferencias (como los tamaños de las instancias, o la opción de tener múltiples compartimientos en los camiones). Se han usado tanto técnicas completas, las cuales han demostrado ser viables para instancias pequeñas a medianas, algoritmos basados en técnicas completas y técnicas incompletas, como lo son *hill-climbing* e *iterated local search*. Estas últimas han demostrado mejores resultados que las técnicas completas.

Una variante a considerar para investigaciones futuras podría ser la incorporación de múltiples plantas de procesamiento de leche en lugar de una única planta central.

De acuerdo a la introducción que se hizo, entregar afirmaciones RELEVANTES basadas en los experimentos y sus resultados. Incluir: Conclusiones sobre el problema, análisis de los resultados, análisis de la técnica usada, qué falló, qué se podría mejorar, trabajo futuro que se podría realizar.

## 10. Bibliografía

### Referencias

- [1] Massimiliano Caramia and Francesca Guerriero. A milk collection problem with incompatibility constraints. *Interfaces*, 40(2):130–143, 2010.
- [2] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6:80–91, 10 1959.
- [3] Dooley, AE, WJ, Parker, and HT Blair. Modelling of transport costs and logistics for on-farm milk segregation in new zealand dairying. *Computers and electronics in agriculture*, 48(2):75–91, 2005.
- [4] El Fallahi, Abdellah, Christian Prins, and Roberto Wolfler Calvo. A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem. *Computers & Operations Research*, 35(5):1725–1741, 2008.
- [5] Leonardo Junqueira, José F. Oliveira, Maria Antônia Carravilla, and Reinaldo Morabito. An optimization model for the vehicle routing problem with practical three-dimensional loading constraints. *International Transactions in Operational Research*, 20(5):645–666, 2013.
- [6] Shuguang Liu, Lei Le, and Sunju Park. On the multi-product packing-delivery problem with a fixed route. *Transportation Research Part E: Logistics and Transportation Review*, 44(3):350–360, 2008.
- [7] Germán Paredes Belmar, Armin Lüer-Villagra, Vladimir Marianov, Cristián Cortés, and Andrés Bronfman. The milk collection problem with blending and collection points. *Computers and Electronics in Agriculture*, 134:109–123, 01 2017.
- [8] Germán Paredes Belmar, Vladimir Marianov, Andrés Bronfman, Carlos Obreque, and Armin Lüer-Villagra. A milk collection problem with blending. *Transportation Research Part E: Logistics and Transportation Review*, 94:26–43, 10 2016.
- [9] Jayaram K Sankaran and Rahul R Ubgade. Routing tankers for dairy milk pickup. *Interfaces*, 24(5):59–66, 1994.
- [10] Constanza Andrea Soto Caviedes. Un acercamiento meta-heurístico para el problema de recolección de leche con selección y mezcla. *Memoria de titulación, UTFSM*, 10 2019.
- [11] Éric Taillard. Parallel iterative search methods for vehicle routing problems. *Networks*, 23(8):661–673, 1993.
- [12] JORGE ANDRÉS VILLAGRÁN MUÑOZ. Acercamientos basados en búsqueda local para el problema de recolección de leche con mezcla. *Memoria de titulación, UTFSM*, 10 2019.