Proyecto Rappi Flujo en Redes

Paula Artunduaga[1], Iván García[2]

[1] Universidad de los Andes
Código 201715960
pa.artunduaga@uniandes.edu.co

[2] Universidad de los Andes
Código 201614488
if.garcia@uniandes.edu.co

1 Introducción

Cada día la competitividad entre las empresas crece, donde aquellas que sobreviven en el mercado es gracias a que han logrado adaptar sus cadenas de suministro a las necesidades de los usuarios. Ahora bien, un aspecto fundamental en las cadenas de suministro es la forma en la que los consumidores reciben los productos terminados. Aún más importante, para la industria de alimentos y comidas rápidas, el transporte y entrega de los productos es un reflejo del servicio al cliente, y por tanto deben buscar las mejores estrategias que ayuden a tomar las mejores decisiones para tener un modelo de entrega de pedidos que sea seguro, rápido y preciso. A raíz de estas estrategias se han desarrollado nuevas formas de negocios, debido que suele ser más barato y eficiente delegar la logística de entrega de bienes tanto perecederos como imperecederos a empresas externas que se dediquen exclusivamente al perfeccionamiento de este proceso. Empresas como Rappi han avanzado en este nuevo mercado, haciendo uso de plataformas tecnológicas y algoritmos de ruteo que combinan a su vez personal y equipo de transporte con el objetivo de trasladar los productos y brindar a un costo mínimo el servicio de domicilios.

Este documento contiene una guía de desarrollo de solución a un problema logístico y operativo para dicha empresa multinacional de plataforma de contacto y transporte de mercancías. El artículo presenta los resultados del estudio de un problema de ruteo de vehículos para 68 instancias en diferentes ciudades del mundo, donde cada una cuenta con un número determinado de órdenes que deben ser cumplidas por una cantidad determinada de rappitenderos en el espacio. Para su solución, se utilizó la herramienta de Google 'OR-Tools' al ser un código abierto de optimización para problemas difíciles de flujo y ruteo de vehículos con restricciones. Para mejorar el método se inicializaron las rutas mediante la heurística de inserción paralela más barata, y posteriormente se utilizó la búsqueda local guiada, una metaheurística de búsqueda local, con el fin de intensificar la búsqueda en la vecindad de soluciones óptimas encontradas. Finalmente, se presentan los resultados obtenidos, evaluando las rutas creadas y órdenes atendidas, teniendo en cuenta que este proceso debe ser evaluado en un límite de tiempo de aproximadamente 1 minuto en el día a día de la compañía.

La presente investigación se enfocó en el uso de la metaheurística Búsqueda Local Guiada y el documento está organizado de la siguiente manera: en la sección 2 se define el problema de ruteo de vehículos VRP y sus variantes; en la 3 se describe el método a utilizar para dar solución al problema; en la 4 se detalla el diseño de experimentos realizado y los resultados obtenidos, y, por último, en la 5 se presentan las conclusiones de la investigación.

2 Descripción del problema y estado del arte

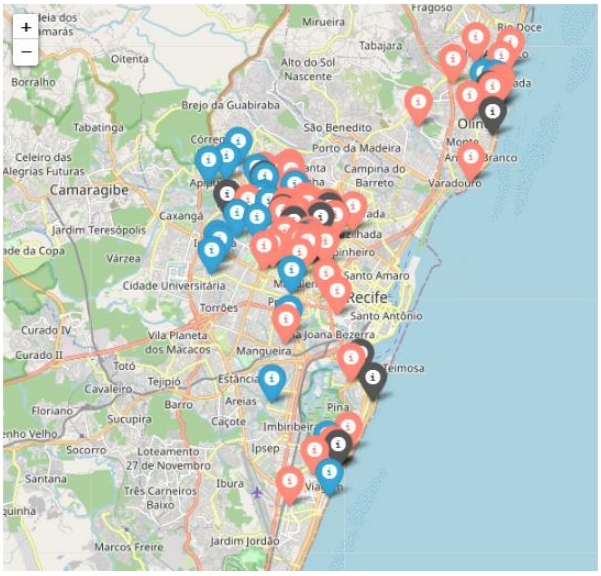
2.1 Descripción del problema de investigación

Rappi necesita de una mensajería que se ajuste a las ideas de flexibilidad, bajo costo y respuesta rápida, lo cual requiere de un esfuerzo importante en la planeación, el control y la comunicación. Uno de los sub-problemas más relevantes de esta actividad, y que más incide en la eficiencia del servicio, es la

definición de rutas para los mensajeros. Este problema es una variante del problema de ruteo de vehículos básico VRP: Open Capacitated Vehicle Routing Problem with Time Windows and Pickups & Deliveries (OCVRPTWPD), ya que existen un conjunto de solicitudes de transporte que se conocen de antemano y que tienen que ser satisfechas por una determinada flota de vehículos (que pueden ser domiciliarios en moto, carro, bicicleta o sin ningún tipo de vehículo). Cada solicitud se caracteriza por su lugar de recogida (origen-tienda), su lugar de entrega (destino-usuario) y los rappitenderos disponibles en el momento que estén clasificados como aptos para atender dicha orden. Para cada lugar de recogida y entrega, se especifican también una ventana de tiempo, ubicación geográfica y tiempos de servicio. La capacidad para atender máximo 3 órdenes, la longitud máxima de su intervalo de funcionamiento (tiempo límite que puede tener una ruta) y una ubicación geográfica de inicio se dan para cada domiciliario. Para cumplir con las solicitudes, se debe planificar un conjunto de rutas de manera que cada solicitud sea transportada desde su origen hasta su destino exactamente una vez (aunque puede pasar que algunas ordenes no sean posible de satisfacer y se cobre por ello una penalización).

Ahora bien, se debe encontrar una solución para un total de 68 instancias, las cuales están conformadas individualmente por $'n'$ ordenes que deben ser atendidas y $'r'$ rappitenderos para atender dichos pedidos. A continuación, se muestra una imagen de todos los nodos existentes para la instancia más pequeña: Instancia "3a1e169b8". Esta instancia posee 22 órdenes (22 clientes y 14 tiendas) y 53 rappitenderos:

Figura 1: Instancia "3a1e169b8": Rojo: Ubicación inicial de cada rappitendero. Gris: Ubicación de las tiendas. Azul: Ubicación de los usuarios.



El Problema de Ruteo de Vehículos es un nombre genérico que se da a una clase muy extensa de problemas consistentes en encontrar rutas óptimas de reparto desde uno o varios nodos de inicio generalmente llamados depósitos a un determinado número de ciudades o clientes de manera que se satisfagan ciertas restricciones. De forma más general, los problemas de generación de rutas de vehículos industriales rara vez son puros y, a menudo, presentan restricciones complicantes:

- VRP: se busca asignar algunos modos de transporte para satisfacer algunas rutas con el fin de dar entrega de un producto o realizar un servicio a un conjunto de clientes en el espacio.
- CVRP: existe una capacidad de carga máxima para los vehículos en cada ruta. Este límite de carga puede verse reflejada en varios factores, como el tiempo máximo de ruta o el número de paradas que se pueden realizar.
- OVRP: Los vehículos usados en las rutas no se encuentran en la obligación de comenzar y terminar en un nodo común (no existe depósito), al contrario, cada domiciliario se encuentra distribuido por el espacio geográfico de un territorio.

- VRPTW: los nodos donde para un vehículo en una ruta tienen ventanas de tiempo dentro de las cuales deben realizarse las entregas.
- VRPTWPD: difiere del VRPTW por las restricciones de precedencia adicionales, es decir, las restricciones que indican que el origen de cada pedido debe visitarse antes del destino correspondiente. En esta variante, es necesario mover los artículos desde ciertos puntos de recogida a otros puntos de entrega.

2.2 Estado del arte del problema

El problema de rutas de vehículos es un problema de programación entera que también se encuadra dentro de la categoría de los problemas NP-Hard. Para este tipo de problemas, cuando el tamaño de este es excesivamente grande, es deseable obtener soluciones aproximadas que puedan ser obtenidas con una rapidez relativa y que sean lo suficientemente parecidas a la solución óptima. Generalmente, la literatura sobre VRP se centra en este tema: encontrar una solución exacta que no requiera tanto esfuerzo computacional o encontrar una solución aproximada que, en un tiempo razonable, dé lugar a soluciones aceptables.

2.2.1 Métodos exactos:

De primera mano, la mayoría de los VRP y sus variantes se modelan como problemas de programa lineal entero mixto (MIP) con variables de decisión de dos índices o rara vez con tres índices para el flujo de diferentes tipos de vehículos [10]. A continuación, se presenta un ejemplo de un modelo MIP que busca representar y dar solución al problema anteriormente descrito en la sección 2.1:

- **Contextualización - Definición de nodos y arcos:**

En primer lugar, se tienen en cuenta la existencia de diferentes órdenes, que se dividen respectivamente en una parada de pickup (Store) y una de delivery (User) establecidas en el espacio que deben ser visitadas. Se debe considerar además que distintas órdenes pueden compartir la misma ubicación geográfica en la parada de pickup, es decir, **es posible tener varios nodos Store para una misma ubicación geográfica**. De acuerdo con la información anterior se tendrán:

$$S \rightarrow Conjunto\ de\ Stores\ (conjunto\ de\ órdenes\ a\ ser\ atendidas)$$
$$S = \{IDOrden_1, IDOrden_2, \dots, IDOrden_n\}$$
$$U \rightarrow Conjunto\ de\ Users$$
$$U = \{n + IDOrden_1, n + IDOrden_2, \dots, n + IDOrden_n\}$$

Teniendo entonces el conjunto de paradas que deben realizar los rappitenderos como:

$$N = U \cup S$$

Así como el conjunto de rappitenderos:

$$R \rightarrow Conjunto\ de\ rappitenderos\ disponibles\ en\ la\ instancia$$
$$R = \{IDRappitendero_1, IDRappitendero_2, \dots, IDRappitendero_m\}$$

Donde θ_k representará el nodo de inicio de cada rappitendero, y por facilidad θ será un nodo ficticio que recogerá como un "depósito" a todos los rappitenderos, tanto los rappitenderos que no se incluyan en ninguna ruta, como aquellos que terminen la recogida y entrega de órdenes en esa instancia. Ahora bien, se debe tener presente que gracias a condiciones y variantes externas al problema no todos los rappitenderos pueden atender todas las órdenes, solo aquellas a las que tiene la característica de ser prospecto, ocasionando que cada rappitendero $k \in R$ tenga asociado en el modelo un conjunto de paradas que puede realizar: $\boldsymbol{N_k = U_k \cup S_k}$ tal que $U_k \subseteq U, S_k \subseteq S$, siendo $\boldsymbol{k_i}$ el número de rappitenderos que pueden atender la orden del usuario $i \in U$. Finalmente, también se tendrán los nodos inicio y fin, siendo el grafo por rappitendero el siguiente: $\boldsymbol{G_k = (V_k, A_k)}$, siendo el conjunto de vértices o nodos

$\boldsymbol{V_k = N_k \cup \{\theta_k, \theta\}}$, y A_k el conjunto de arcos factibles para dicho rappitendero $\boldsymbol{A_k = V_k \times V_k}$. Además, existirá un costo de transporte dependiente del tipo de vehículo en el que se transporte un rappitendero, existiendo el conjunto y subconjunto:

$$T: Conjunto\ de\ tipos\ de\ vehiculos\ en\ el\ que\ se\ puede\ transportar\ un\ rappitendero$$
$$T = \{Car, Motorcycle, Bycicle, Walking\}$$
$$T_k \rightarrow Vehículo\ en\ el\ que\ se\ transporta\ el\ rappitendero\ \ k \in R$$

A continuación, se presenta un esquema general de los nodos y arcos del sistema $G(V, A)$:

- **Nodos para la ubicación inicial de los rappitenderos:**

Para cada rappitendero disponible, existe un nodo que representa su ubicación en el mapa una vez obtenida la instancia.

- **Nodos de las tiendas:**

Existe un nodo por cada tienda en la que se tiene por lo menos una orden que debe ser recogida.

- **Nodos de los usuarios:**

Existe un nodo por cada usuario que haya realizado una orden al momento de registrar la instancia.

- **Nodo "ficticio" final de rappitenderos:**

Existe un nodo dummy θ que funciona como depósito compartido para recoger a los rappitenderos, tanto los que terminan su ruta como lo que nunca fueron asignados a ninguna orden.

- **Arcos ubicación inicial rappitenderos – Stores:**

Existe un arco de un rappitendero a una tienda únicamente para aquellos que sean prospectos de cumplir el pedido a recoger en la tienda.

- **Arcos Stores – Usuarios:**

Existe un arco que representa el desplazamiento de una tienda a un usuario, teniendo en cuenta los pedidos a entregar.

- **Arcos Stores – Stores:**

Existe un arco que representa el desplazamiento entre dos tiendas, teniendo en cuenta los rappitenderos prospectos.

- **Arcos Usuarios – Usuarios:**

Existe un arco que representa el desplazamiento entre dos usuarios, teniendo en cuenta de que, para entregar un pedido es necesario primero hacer el pickup y luego el delivery.

- **Arcos Rappitenderos – Nodo ficticio Rappitenderos:**

Existe un arco por cada ubicación inicial del rappitendero al nodo ficticio de llegada de los rappitenderos, con objeto de recibir a los rappitenderos que en una instancia determinada no atiendan ninguna orden.

- **Arcos Usuarios - Nodo ficticio Rappitenderos:**

Existe un arco por cada ubicación de entrega de una orden nodo ficticio de llegada de los rappitenderos, con objeto de recibir a los rappitenderos que hayan completado su ruta debido a limitaciones de tiempo y capacidad.

- **Formulación del modelo:**

Ya definidos los arcos y nodos necesarios para el modelo, se puede generar la formulación.

- *Variables de decisión:*

$$x_{i,j,k}: \begin{cases} 1, si\ el\ rappitendero\ k \in R\ atraviesa\ el\ arco\ (i,j) \in A_k \\ 0, d.l.c \end{cases}$$

$$y_i \begin{cases} 1, si\ el\ pedido\ del\ Stores\ i \in S\ es\ atendido\ por\ algún\ rappitendero \\ 0, d.l.c \end{cases}$$

$$w_{i,k}: Tiempo\ en\ el\ que\ el\ rappitendero\ k \in R\ comienza\ el\ servicio\ en\ el\ nodo\ i \in V_k$$

- *Parámetros:*

$$s_i: Tiempo\ de\ servicio\ gastado\ en\ el\ nodo\ i \in V$$

$$a_i: Tiempo\ mínimo\ en\ el\ que\ el\ puede\ llegar\ un\ rappitendero\ a\ la\ parada\ i \in V$$

$$b_i: Tiempo\ máximo\ en\ el\ que\ el\ puede\ llegar\ un\ rappitendero\ a\ la\ parada\ i \in V$$

$$h = 3 \rightarrow\ Max\ número\ de\ órdenes\ que\ puede\ atender\ cada\ rappitendero$$

$$r \rightarrow Número\ total\ de\ rappitenderos\ en\ la\ instancia$$

- *Restricciones:*

1. Si una tienda no es visitada, se agrega a las órdenes que no fueron visitadas.

$$\sum_{k \in k_i} \sum_{j \in N_k} x_{i,j,k} + (1 - y_i) = 1 \quad ; \forall\ i \in S$$

2. Un nodo delivery (User) debe ser visitado en el caso de que el nodo pickup (Store) correspondiente también lo haya sido, y que por ambos nodos pase el mismo rappitendero.

$$\sum_{j \in V_k} x_{i,j,k} - \sum_{j \in V_k} x_{j,n+i,k} = 0 \quad ; \forall\ k \in R, i \in S_k$$

3. Restricción de balance para nodos de transbordo (nodos Store y Users):

$$\sum_{j \in V_k} x_{i,j,k} - \sum_{j \in V_k} x_{j,i,k} = 0 \quad ; \forall\ k \in R, i \in N_k$$

4. Restricción de balance para los nodos de Rappitenderos:

$$-\sum_{j \in S_k} x_{\theta_k,j,k} = -1 \quad ; \forall\ k \in R$$

5. Restricción de balance para el nodo ficticio que recoge al final del modelo el flujo de todos lo rappitenderos:

$$\sum_{i \in U_k \cup \{\theta_k\}} x_{i,\theta,k} = r \quad ; \forall\ k \in R$$

6. Se da valores coherentes a la variable de inicio del tiempo de servicio en cualquier nodo en la ruta de un rappitendero específico, teniendo en cuenta el tiempo de servicio en dicho nodo, y el tiempo de transporte que depende del tipo de vehículo que use ese rappitendero.

$$w_{i,k} + s_i + t_{i,j,T_k} - w_{j,k} \leq \left(1 - x_{i,j,k}\right)M \quad ; M \gg 0,\ \forall k \in R, (i,j) \in A_k$$

7. Si un nodo va a ser visitado, su servicio debe comenzar entre las ventanas de tiempo especificadas.

$$a_i \sum_{j:(i,j) \in A_k} x_{i,j,k} \leq w_{i,k} \leq b_i \sum_{j:(i,j) \in A_k} x_{i,j,k} \quad ; \forall\ k \in R, i \in V_k$$

8. Para una misma orden que será ejecutada, el nodo pickup (Store) debe ser atendido antes del nodo delivery (User).

$$w_{i,k} \leq w_{n+i,k} \quad ; \forall\ k \in R, i \in S_k$$

9. Un rappitendero puede atender máximo tres órdenes dentro de su ruta (visitar máximo 3 tiendas).

$$\sum_{j \in S_k} \sum_{i \in V_k} x_{i,j,k} \leq h \quad ; \forall\, k \in R$$

10. Naturaleza de las variables:

$$x_{i,j,k} \in \{0,1\} \quad ; \forall\, k \in K, (i,j) \in A_k$$
$$y_i \in \{0,1\} \quad ; \quad \forall\, i \in S$$
$$w_{i,k} \geq 0 \quad ; \forall\, k \in K, i \in V_k$$

- *Función Objetivo:*

$$Minimizar\ los\ tiempos\ totales\ de\ ruta$$
$$\min \sum_{k \in K} w_{\theta,k} + 3600 * \sum_{i \in S} (1 - y_i)$$

Existen varios solucionadores disponibles, tanto comerciales (LINGO, CLPEX) y gratuitos (SCIP, LP_Solve), que utilizan métodos MP como simplex, branch & bound, branch & price, etc. Por ejemplo, Dell'Amico et al [7], publicaron en el año 2005 un extenso estudio de solución para un problema VRPSPD por medio de métodos exactos, haciendo uso de un modelo de Branch & Price por medio de, por un lado, programación dinámica exacta, y por otro lado, el uso de relajación de los espacios. No obstante, dada las características de complejidad de un VRP y sus variantes, tal como en el caso de investigación del VRPSPD esta forma de solución resulta ineficiente, especialmente cuando la cantidad de instancias que se pretenden solucionar es relativamente grande y algunas de dichas instancias presentan un número de datos considerablemente alto.

2.2.2 Métodos no exactos:

Dada la complejidad del OCVRPTWPD como un problema de optimización combinatoria, se ha vuelto de interés el desarrollo de algoritmos que permitan obtener una buena solución (no necesariamente la solución óptima), pero que en su lugar logren disminuir el tiempo computacional al momento de resolver problemas de gran complejidad. Los problemas de generación de rutas para vehículos conducen a formulaciones desafiantes que requieren el desarrollo de estrategias de solución sofisticadas y motiva el diseño de heurísticas y metaheurísticas inteligentes. Ahora bien, las heurísticas generalmente estudiadas pueden clasificarse en 3 categorías: heurísticas constructivas, heurísticas de dos fases y heurísticas de mejora iterativa [11].

En la primera clasificación se encuentran aquellos algoritmos que buscan encontrar primeras soluciones factibles relativamente buenas, tales como Clarke and Wright, inserción menor costosa, Fisher y Jaikumar, entre otros. Luego, en las heurísticas de dos fases ocurre que el problema busca dividirse en dos columnas que se sostienen la una a la otra. Para trabajar bajo el lema de 'divide y vencerás' han existido diferentes ideas a lo largo del tiempo, por ejemplo, Bodin y Sexton (1986) propusieron un procedimiento iterativo que, en cada iteración, primero (re) asignan solicitudes a los diferentes vehículos y luego resuelven un PDPTW para cada vehículo [3]. Otras investigaciones más recientes muestran los avances en el área y su combinación con métodos lineales (exactos) para lograr el propósito de buenos resultados en un tiempo razonable. Con el fin de solucionar un problema de ruteo de mensajeros en motocicletas con ventanas de tiempo (VRPTW), Arboleda et al proponen en 2018 un método híbrido de dos fases: la primera, usa una técnica de agrupación que asigna los clientes a una determinada ruta, y la segunda, por medio de un modelo de programación lineal entero mixto, establece la secuencia de dichas rutas. De este experimento, donde en la primera fase se evalúan dos heurísticas constructivas (heurística de los ahorros y la heurística del barrido) y en la segunda se realiza un modelo exacto, se obtiene que mejora considerablemente la duración total de la ruta, aunque en algunos casos, aumenta la distancia recorrida por el vehículo [2].

La tercera categoría de las heurísticas es aquella que busca, una vez se tiene una solución previa al problema de ruteo en cuestión, intentar mejorar algún procedimiento de búsqueda local. Algunas de estas heurísticas son: GENI y GENIUS, transferencias cíclicas, intercambios r-opt, mecanismos unstringing y stringing (US) [16]. A modo de ilustración, con el fin de mejorar la calidad de las soluciones obtenidas mediante una heurística de construcción, Van der Bruggen et al. (1993) proponen un procedimiento de mejora local para el PDPTW de un solo vehículo basado en intercambios de arco siguiendo el procedimiento de 'búsqueda de profundidad variable' para el problema de recogida y entrega de un solo vehículo con ventanas de tiempo [4].

Finalmente, existe otro tipo de algoritmos dedicados, entre otros campos, a resolver de forma más o menos rápida y robusta los problemas de ruteo de vehículos: las técnicas metaheurísticas. '*Los metaheurísticos son métodos aproximados diseñados para resolver problemas de optimización combinatoria, en los que los heurísticos clásicos no son efectivos. Los metaheurísticos proporcionan un marco general para crear nuevos algoritmos híbridos, combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los mecanismos estadísticos'* [11]. Técnicas como la Optimización por Espiral, la Colonia de Hormigas, el paradigma de Búsqueda Tabú y la Búsqueda Local Guiada han sido probadas en el campo de la resolución de VRP y sus variantes, obteniendo el mejoramiento de soluciones previas por medio de otros métodos.

3 Metodología

3.1 Esquema de solución

Lo que se quiere resolver es un problema OCVRPTWPD, donde se debe elegir qué rappitendero va a atender qué pedidos, y de dichos domicilios que logran ser asignados, en qué orden van a ser atendidos dentro de la ruta correspondiente, minimizando de forma simultánea el tiempo total de recorrido. Ahora bien, existen instancias en ciudades del mundo con muy poca oferta y demanda (ver figura 1, sección 2.1) con solamente unos cuantos rappitenderos y unas pocas ordenes, así como instancias en ciudades principales donde la empresa debe decidir el ruteo y asignación de entre 800 domiciliarios a más de 50 órdenes. Con esto en mente, la estrategia utilizada para reducir la cantidad de nodos y arcos a resolver es tratar de reducir el problema grande (una instancia a priori) en problemas más pequeños. A continuación, se relata un esquema de cómo se realiza esta partición:

3.1.1 Reorganización en clústeres:

Se eligió realizar una clusterización de los datos para cada una de las instancias a evaluar por medio de la técnica k-means. Este es un algoritmo basado en particiones, donde no existe previamente un aprendizaje supervisado, es decir, no existen clases predefinidas para realizar la segmentación. *'Se utiliza para dividir los casos o las variables de un conjunto de datos en grupos o conglomerados que no se superponen, según las características descubiertas'*[9]. En este caso la característica escogida para desarrollar los clusters es una variable continua: la distancia euclidiana entre los diferentes puntos (que en este caso son los nodos de las ubicaciones de las tiendas, los usuarios y los rappitenderos) a unos puntos denominados centroides. Lo que se busca con esta forma de clusterización es encontrar grupos de puntos de forma iterativa que sean similares entre sí y diferentes a los puntos pertenecientes a otros grupos de acuerdo con la característica seleccionada.
La forma en la que se está operando el algoritmo es la siguiente para cada instancia:
1. Se definen los posibles clústeres a realizar en la instancia. Para ello se evalúa un rango posible de 2 a 8 clusters, donde para cada uno se generan de manera aleatoria los centroides en el espacio (dependiendo de la cantidad de clusters que se están desarrollando en el momento) y se clasifica cada punto de interés en el problema original en alguno de los clusters creados según la característica seleccionada, en este caso quedando asignado al clúster cuyo centroide se encuentre a una menor distancia euclidiana. Este proceso se repite hasta que las agrupaciones no cambien entre iteraciones consecutivas.

2. Se decide el tipo de clúster que se realizará en la instancia. Existen dos métodos que se tienen en cuenta para escoger el mejor clúster de los evaluados en el paso 1. El primer método, es a través de la métrica Score, ya que permite evaluar la eficiencia de la división de los clústeres al calcular el error cuadrático medio entre la distancia entre cada punto a su centroide teniendo en cuenta la cantidad de clústeres. De acuerdo con esta métrica, se selecciona el punto en el que el error cuadrático medio cambia su comportamiento exponencial a comportamiento logarítmico. No obstante, para el problema de Rappi, que busca también lograr atender la mayor cantidad de ordenes posibles, como segundo método se mira cuántas ordenes se pueden cumplir en el mejor de los casos dependiendo de la cantidad clústeres por la que se está dividiendo. Este dato se define como cuántos conjuntos tienda-usuario que definen una orden quedan por dentro de los clústeres establecidos, pues si alguna tienda queda por fuera del clúster de un usuario o viceversa, la orden se asume como imposible de cumplir.

A continuación se presenta un ejemplo de los clusters obtenidos para la instancia "3a1e169b8":

Figura 2: Instancia "3a1e169b8". Curva Score para la elección del número de clústeres a realizar. Tendencia a 3 clústeres.

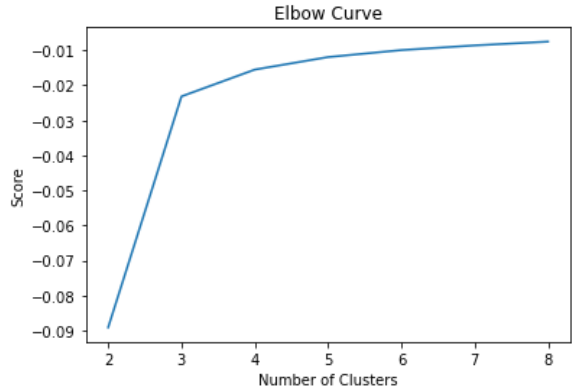


Figura 3: Instancia "3a1e169b8". Evaluación de la cantidad de órdenes que podrían satisfacerse a priori según la cantidad de clusters que se realicen

```
For 2 cluster, could be accomplish 22 orders
For 3 cluster, could be accomplish 21 orders
For 4 cluster, could be accomplish 8 orders
For 5 cluster, could be accomplish 5 orders
For 6 cluster, could be accomplish 5 orders
For 7 cluster, could be accomplish 4 orders
For 8 cluster, could be accomplish 4 orders
The best possible solution is: 2 clusters, could accomplish 22 orders in current instance
```

Figura 4: Instancia "3a1e169b8". Resultado técnica de clusterización *k-means:* 2 clusters (azul y rojo).

3.1.2 Separación del problema por el tipo de vehículo:

Con la clusterización por k-means, ahora se resolverá un modelo por cada clúster creado (2 modelos por ejemplo para la instancia "3a1e169b8"). Estos clústeres ayudan a reducir la complejidad del problema y mejorar la potencia de los algoritmos de solución al resolver problemas más pequeños. No obstante, se determinó tratar la información en segmentos aún más separados, teniendo en cuenta el tipo de vehículo que puede tener un domiciliario. Se decidió que en cada clúster se corriera el algoritmo de resolución de problemas de ruteo de Google haciendo diferencia para el tipo de vehículo 'moto', 'carro' y 'bicicleta'. Es decir, para el ejemplo de la instancia "3a1e169b8", se toma el primer clúster y se encuentra la solución de rutas para los domiciliarios en moto, luego se repite el proceso para los domiciliarios en carro y finalmente en bicicleta. Por último, si llegan a presentarse órdenes que hayan quedado asignadas a dos o más rutas que pertenecen a vehículos distintos, se evalúa manualmente cual es la mejor (con cual se genera el menor tiempo total en ruta). Este proceso se repite para todos los clusters que existan en una instancia.

3.1.3 Herramienta OR-Tools, heurística de la Inserción Paralela más Barata y metaheurística de Búsqueda Local Guiada:

Para encontrar la solución a los pequeños segmentos del problema original se decidió hacer uso de la herramienta de Google 'OR – Tools'. Este es *'un paquete de software de código abierto para optimización, diseñado para abordar los problemas más difíciles del mundo en el enrutamiento de vehículos, flujos, programación lineal y entera y programación de restricciones'* [8]. La ventaja de usar OR-Tools radica en que busca encontrar la mejor solución (no necesariamente óptima) entre las miles de posibles de soluciones que puede tener un problema de ruteo de vehículos, por medio de la ejecución de algoritmos robustos que provoquen la reducción del conjunto de búsqueda. Existe toda una documentación de Google para resolver desde problemas de flujo en redes puros como el problema de flujo máximo, así como la forma de programar la solución a problemas de asignación, planificación y enrutamiento (ejemplos para la resolución del problema del viajante, el VRP y sus diferentes variantes).

A esta herramienta se le pasan por parámetros 4 cosas: primero, una matriz con números enteros la cual representa los costos de ir de un nodo de interés a otro; segundo, se le ingresan las restricciones complicantes del modelo como la capacidad de un domiciliario, las ventanas de tiempo permitidas en cada nodo, la obligación de visitar un nodo de pick up antes de un nodo delivery para una misma orden, así como la función objetivo teniendo en cuenta las penalizaciones por cada orden que no logre ser satisfecha. Luego, se le ingresa al solucionador la estrategia que debe implementar para generar una primera solución al modelo. En este caso se le ingresa como primera estrategia de solución la heurística de Inserción Paralela más Barata, la cual *'construye iterativamente una solución insertando el nodo más barato en su posición más barata; el costo de inserción se basa en la función de costo del arco'* [14]. Y, por último, se incluye la estrategia que el solucionador debe usar para encontrar una mejor solución

(mejorar la solución inicial encontrada con la heurística). Para ello se usó una estrategia de búsqueda más avanzada, llamada Búsqueda Local Guiada , metaheurística que permite al solucionador escapar de un *mínimo local,* una solución que es más corta que todas las rutas cercanas, pero que no es el mínimo global.

4 Resultados y análisis

El modelo fue ejecutado en un computador con características Intel Core i5-9300H CPU @ 2.40Ghz 8GB Ram y sistema operativo de 64 bits y la metodología fue codificada en lenguaje Python en un Jupyter Notebook. Después de correr las 68 instancias del modelo, se tiene el siguiente resumen de los resultados:

Tabla 1: Solución obtenida para 37 de las 68 instancias.

	instance_id	num_orders	num_couriers	matched_orders	total_routes_time	avg_route_time	runtime
1	3a1e169b8	22	53	10	56984	1378.4	78.12337255
2	a5d7b3b66	10	44	7	17466	952.2857143	62.87607217
3	dd1dbcc27	11	138	7	22626	1175.142857	198.951961
4	016141811	25	31	10	67034	1303.4	260.0964131
5	2cfb2a25d	15	41	14	18111	1036.5	306.1361134
6	99a5d9f53	11	166	8	19272	1059	139.8096917
7	90e425904	44	172	4	39207	801.75	276.2956202
8	c8f092b66	17	326	14	27113	1165.214286	488.1184425
9	328b4c0a9	15	116	10	28509	1050.9	549.6788542
10	2d759ab1b	15	137	7	35237	919.5714286	641.3245189
11	282eb26ab	10	146	8	14665	933.125	717.6616971
12	5e073964f	13	246	7	29357	1108.142857	884.5463789
13	d0c903392	21	214	21	10179	484.7142857	1021.568562
14	e973a0ed0	18	135	11	36334	1012.181818	1113.46507
15	d15ca2042	10	190	4	25773	1043.25	1190.086893
16	0a7ac04ce	16	255	13	25247	1111.307692	1282.041077
17	79fc6e4b1	11	271	8	18768	996	1359.253603
18	674d0e30e	11	191	3	30663	621	1451.313608
19	57e549baa	11	416	6	23538	923	88.96545482
20	039812c88	13	332	12	5343	145.25	119.6004601
21	31a5323b9	15	325	6	37713	885.5	276.8506818
22	21c1b3e52	14	477	8	31115	1189.375	473.6110699
23	bd3821c33	36	479	27	59449	1001.814815	605.2895546
24	7b9668b8a	60	690	46	105119	1189.543478	799.5275037
25	b3ebe98a3	41	203	33	33558	144.1818182	990.8151176
26	7636e8513	29	609	21	47167	874.6190476	1268.107553
27	99a10db5d	15	421	8	33136	992	1393.346867
28	642246e6f	13	604	12	15001	950.0833333	3014.496821
29	3fc9f4c81	25	803	20	35207	860.35	2829.199116
30	a3fb6cd9b	35	695	26	58666	1010.230769	3171.281461
31	9810f86e5	20	1294	27	31846	246.1481481	1485.508446
32	a366cc689	22	480	15	40224	1001.6	3416.651237
33	7bcb8cc32	47	613	29	95848	1070.62069	1649.876446
34	6d41d2d0f	25	469	10	61228	722.8	2638.843035
35	050314e7f	293	2377	250	202374	190.296	2138.737007
36	19df4db80	590	6599	393	722246	33.19592875	1416.021
37	9fe226dcc	38	744	38	3122	82.15789474	1820.039394

De los resultados arrojados se tiene que en promedio el 67% del total de las ordenes existentes en cada instancia son atendidas, con un tiempo promedio de ruta (sin contar el tiempo adicional por las penali-

zaciones de ordenes no completadas) de 901.41 segundos o 15.02 minutos por orden realizada. Lo anterior refleja un buen resultado, pues el objetivo principal del modelo es obtener el mínimo tiempo total de ruta en la operación de la empresa. No obstante, el porcentaje de ordenes que no son atendidas podría ser justificado por la perdida de datos al realizar los clústeres o al no considerar dentro del modelo rappitenderos sin ningún tipo de vehículo que en realidad podrían atender en el mejor de los casos a las ordenes penalizadas actualmente.

5 Conclusiones

Finalmente, luego de varias pruebas experimentales podemos llegar a la conclusión de que la clusterización del espacio de soluciones mediante el algoritmo de k-medias ayuda no solo a mejorar el tiempo de cómputo, reduciendo la complejidad exponencial de manejar todo el problema al mismo tiempo, también permite converger a una solución óptima de una mejor manera, ya que inicialmente ayuda a descartar rutas que son muy difíciles de cumplir en un principio dada la distancia que a la que está la orden de su tienda o de algún rappitendero que pueda cumplirla. En adición, k-medias me permite generar soluciones factibles iniciales mucho más cercanas al optimo ya que estarán desde un principio restringidas al estar en el mismo clúster generado por el algoritmo.

Por otro lado, observando el patrón del tiempo de ejecución, el valor de la función objetivo y la cantidad de ordenes atendidas y rechazadas en cada instancia se puede concluir que a pesar de que la correlación tiende a ser positiva entre todos los factores previamente mencionados, hay instancias que al verse ajustadas a restricciones complicantes tan estrictas como las ventanas de tiempo tienden a tener una gran cantidad de indicadores perjudiciales, como mayor cantidad de ordenes rechazadas. A pesar de que es posible que la solución arrojada por el modelo no sea la más optima, como consultores de Rappi nuestra propuesta se enfoca en encontrar la mejor solución posible en un rango de tiempo ajustable a los requerimientos operacionales de Rappi. Si bien es cierto que las instancias más altas tienen un tiempo de ejecución moderado, quisimos mantenernos entre un punto medio de lo mejor que puede hacer el modelo y el tiempo esperado por Rappi como organización para poder solucionar el problema.

Bibliografía

[1]Aguilar, J. (2017). Resolución computacional de un problema de optimización combinatorio hibrido. *Ciencia e Ingeniería, 38*(2). Obtenido de http://www.redalyc.org/articulo.oa?id=507555007001

[2] Arboleda, J. J., Heredia, A. D., & Orejuela, J. P. (Enero-Junio de 2018). Método de dos fases para el problema de ruteo de mensajeros en motocicleta con ventanas de tiempo. *Entramado, 14*(1), 268-281. doi:http://dx.doi.org/10.18041/entramado.2018v14n1.27120

[3] Bodin, L., & Sexton, T. R. (1986). The multi-vehicle subscriber dial-a-ride problem. *TIMS Studies in the Management Sciences , 22*, 73-86. Obtenido de https://www.researchgate.net/publication/247932478_The_multi-vehicle_subscriber_dial-a-ride_problem

[4] Bruggen, L. J., Lenstra, J. K., & Schuur, P. C. (1993). ariable-depth search for the single-vehicle pickup and delivery problem with time windows. *Transportation Science, 27*(3), 298-311. doi:https://doi.org/10.1287/trsc.27.3.298

[5] Burbano, L. (2011). *La tercera parte de la logística: el servicio a domicilio en el sector de comidas rápidas en Cali.* Cali. Obtenido de https://www.usbcali.edu.co/sites/default/files/005_logisticacomidasrapidas.pdf

[6] Cruz, C., & González, L. (2013). *Desarrollo de un algoritmo híbrido para la resolución del problema de ruteo de vehículos con entrega y recogida simultáneas (VRPSPD).* Universidad Industrial de Santander, Facultad de Ingenierías Fisicomecánicas, Bucaramanga. Obtenido de http://tangara.uis.edu.co/biblioweb/tesis/2013/149648.pdf

[7] Dell'Amico, M., Righini, G., & Salani, M. (2005). *A Branch-and-Price Approach to the Vehicle*

Routing Problem with Simultaneous Distribution and Collection. doi:10.1287/trsc.1050.0118

[8]Google.Inc. (s.f.). *Google OR-Tools*. Obtenido de https://developers.google.com/optimization

[9]Morissette, L., & Chartier, S. (2013). The k-means clustering technique: General considerations and implementation in Mathematica. *Tutorials in Quantitative Methods for Psychology, 9*(1), 15-24. doi:10.20982/tqmp.09.1.p015

[10] Nagy, G., Wassan, N. A., Speranza, M. G., & Archetti, C. (2015). The vehicle routing problem with divisible deliveries and pickups. *Transportation Science, 49*(2), 271–294. doi:https://doi.org/10.1287/trsc.2013.0501

[11] Orrego, J. P., Ospina, D., & Toro, E. M. (2016). Solución al Problema de Ruteo de Vehículos con Capacidad Limitada (CVRP) usando una técnica metaheurística. *Scientia Et Technica, 21*(3), 225-233. Obtenido de https://www.redalyc.org/articulo.oa?id=849/84950585004

[12] Osman, I. H., & James P., K. (1996). *Meta-Heuristics: Theory and Applications.* KLUWER ACADEMIC PUBLISHERS. doi:10.1007/978-1-4613-1361-8_1

[13] Vásquez Agudelo, C. A., & Hernández Acosta, R. A. (2015). *Pasos para implementar un servicio a domicilio para las pequeñas empresas en el sector de comidas rápidas.* Medellín. Obtenido de https://repository.udem.edu.co/bitstream/handle/11407/2153/TG_EAG_81.pdf?sequence=1&isAllowed=y

[14] (BRÄYSY, OLLI;GENDREAU, MICHEL). *Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms.* Obtenido de http://cepac.cheme.cmu.edu/pasi2011/library/cerda/braysy-gendreau-vrp-review.pdf

[15] Vélez, M. C., & Montoya, J. A. (Julio-Diciembre de 2007). Metahurísticos: Una alternativa para la solución de problemas combinatorios en adminitración de operaciones. *EIA Escuela de Ingeniería de Antioquia, 8*. Obtenido de http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S1794-12372007000200009

[16] Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2012). *Heuristics for Multi-Attribute Vehicle Routing Problems : A Survey and Synthesis.* doi:10.1016/j.ejor.2013.02.053