UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MATEMÁTICA

MÉTODO DE CLUSTERIZACIÓN PARA EL VRPTW CON CONSIDERACIONES DE BALANCEO DE TIEMPO

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MATEMÁTICAS APLICADAS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MATEMÁTICO

SEBASTIÁN ANTONIO BUSTOS ATALAH

PROFESOR GUÍA:
FERNANDO ORDOÑEZ PIZARRO

PROFESOR CO-GUÍA:
JOSÉ SOTO SAN MARTÍN

COMISIÓN:
TOMÁS MARTÍNEZ MUÑOZ

Este trabajo ha sido parcialmente financiado por
CMM ANID BASAL ACE210010 y CMM ANID BASAL FB210005.

SANTIAGO DE CHILE
2023

RESUMEN DE LA TESIS PARA OPTAR
AL GRADO DE MAGÍSTER EN CIENCIAS
DE LA INGENIERÍA, MENCIÓN MATEMÁTICAS APLICADAS
Y MEMORIA PARA OPTAR AL
TÍTULO DE INGENIERO CIVIL MATEMÁTICO
POR: SEBASTIÁN ANTONIO BUSTOS ATALAH
FECHA: 2023
PROF. GUÍA: FERNANDO ORDOÑEZ PIZARRO

MÉTODO DE CLUSTERIZACIÓN PARA EL VRPTW CON CONSIDERACIONES DE BALANCEO DE TIEMPO

Al momento de resolver el problema de ruteo de vehículos en ciertos contextos prácticos se vuelve deseable tener cierta noción de zonificación del problema, además existen estrategias que están basadas en realizar una clusterización del problema las cuales pueden ser utilizadas en varios contextos.

En esta tesis se aborda el problema de realizar un método de clusterización para el problema del VRPTW donde se tenga en consideraciones de balanceo de tiempo. Se presenta un algoritmo que permite clusterziar teniendo en consideración tanto las restricciones del problema de enrutamiento de vehículos como las de balanceo por tiempo. Lo propuesto consiste en remplazar el costo de una ruta por el costo de un árbol expansor y para incorporar el balanceo se realiza una aproximación del tiempo en ruta, dependiendo del problema utilizando un algoritmo o una regresión lineal de parámetros, y utilizando el rango se estima que tan balanceada es la solución. Realizando búsquedas locales se construyen las clusterizaciones y posterior a rutear cada cluster se vuelve a juntar y se realiza una última búsqueda local sobre la solución del problema de ruteo siguiendo una estructura análoga a la de la clusterización.

Además, se estudia como eliminar aristas de un grafo completo de forma tal que se mantenga el árbol generador de peso mínimo. Se estudian resultados en un contexto determinista y probabilista, en este ultimo se logra demostrar que el árbol generador esta contenido en el grafo de los k-vecinos asintoticamente en la cantidad de nodos con probabilidad 1, en este caso k es variable y depende de n, la función de distribución de las aristas y un parámetro arbitrario. Esto permite reducir la complejidad del algoritmo, se logra disminuir un termino de la expresión de n^2 a $h\sqrt{n^3 \log(n)}$, donde h es un valor que depende de la función inversa de Ackermann.

Todo tiene un principio y todo tiene un final,
cada historia que termina lleva a un nuevo comenzar,
por eso lo que importa es el camino y no el final.

K.F.

Agradecimientos

En primer lugar, agradecer a mi papa y tía quienes estuvieron todos estos años.

Al profesor Fernando quien, sin conocerme, acepto trabajar conmigo y soporto todos mis "lo que dije la semana pasada está mal porque hay un error" y mis ganas de extender por una semana más todo lo que hacíamos para encontrar algo que me dejara conforme.

A toda la gente de CMAT. Sobre todo, a Patito, Jona y Vilches quienes a lo largo de los años estuvieron ahí, sobre todo en los momentos más complejos nunca me abandonaron. Al resto del equipo quienes hacen todos los sábados más agradables. Y a Gloria quien siempre tendrá mi cariño incondicional.

A Pipe, Arie y Nico con quienes formaron parte de incontables matracas y noches de juego. Tristan a quien siempre le podía preguntar alguna tontera y tirar ideas al aire. Matu quien me acompaño siempre durante plan común e incluso después de eso de manera incondicional. Bary quien también estuvo ahí muchos años a veces para matraquear otras para conversar de la vida. Anto y Mapi quien también me ha acompañado todos estos años de universidad.

A Espi quien, incluso antes de enterar a la U, ha soportado todos mis "aun no supero", todas esas conversaciones esporádicas que tenemos (quejándonos de una que otra cosa) y por todo el cariño que me has dado estos años, gracias por todos estos años siempre tendrás mi cariño. Marijo quien, también desde antes de la u, ha escuchado todas mis tonteras, todas esas conversaciones esporádicas y más que una u otra confusión sobre lo que estamos hablando, gracias por seguir por aquí.

A todos los del equipo de Data Science de SimpliRoute quienes me han tratado con amabilidad, cariño y buena onda. En especial a Tomas, quien tuvo que soportar, ver y escuchar todas mis mentiras y las que terminaron siendo no tan mentiras, por las conversaciones tirando ideas, leyendo papers, debugeando, patito de hule, de la vida y muchas más. Sin tu presencia esto no hubiera terminado como lo hizo.

Por último, pero no por eso una pizca de menos importante, gracias por todo Denisse, aun cuando ya no te encuentres aquí, si no fuera por haberte conocido, por todas esas conversaciones en las tardes noches, mis rants de KS, tu cariño, la alegría que me diste yo no hubiera tenido ni las ganas ni la energía de llegar a donde estoy, siempre te recordare y extrañare.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

Dentro de los problemas de optimización, en el mundo de la logística, se encuentra el de enrutamiento de vehículos (VRP por sus siglas en ingles). Este problema es una generalización del problema del vendedor viajero (Travelling Salesman Problem o TSP) que consiste en tener múltiples vehículos (viajeros). En la literatura cuenta con una gran cantidad de variantes: Capacidades, Ventanas horarias, Pick & Delivery, versiones estocásticas, entre otras. En Braekers et al. [1] se presenta un survey con varias variantes del problema.

El VRP, en su versión mas sencilla, consiste en tener un conjunto de vehículos, un punto de partida y un conjunto de puntos, para cada par de puntos se conoce el costo de utilizar ese arco (usualmente la distancia o el tiempo de viaje entre los puntos). Luego lo que se desea es buscar el conjunto de rutas que genere el menor costo total, donde el costo se considera la suma de todos los arcos utilizados y una ruta se entiende como un orden para recorrer los puntos que parte y termina en el punto de partida, además existe la condición extra que un punto (salvo el de partida) puede ser visitado una sola vez.

Para la elaboración de este escrito se trabajó en conjunto a la empresa SimpliRoute, que se dedica al ruteo de vehículos y facilita los algoritmos heurísticos que usa. Dentro de este trabajo se considerara que los resultados de la empresa forman parte del estado de arte en ruteo de vehículos, por tanto se realizaran comparaciones contra sus soluciones.

Es importante mencionar que al momento de llevar este problema a la realidad el criterio de optimalidad clásico, introducido por Dantzig y Ramser en [2], como el costo/largo del viaje se vuelve insuficiente y surge la necesidad de incorporar condiciones extra a la optimalidad y/o restricciones. Dentro de estas condiciones puede surgir la necesidad de balancear *la carga de trabajo*, donde las nociones de *balanceo* y *carga de trabajo* pueden definirse de varias maneras. Para este último pueden considerarse largo en ruta, capacidad, numero de paradas, tiempo en ruta, entre otras.

Lo que se considere como carga puede tener distintos efectos en el problema de ruteo, esto se puede apreciar en Matl et al. [3]. Entre las condiciones más estudiadas en el balanceo de carga se encuentra la asociada a largo en ruta, en Matl et al. [4] se presenta un estudio de cuáles son las consideraciones que se han tomado en varias investigaciones y un análisis sobre las distintas funciones que permiten medir balanceo. A pesar de que se suele estudiar el balanceo asociado al largo en ruta, en Sivaramkumar et al. [5] se presenta un estudio don-

de se muestra que tener consideraciones asociadas a balancear la duración en tiempo de las rutas de un VRP con ventanas horarias (abreviado como VRPTW) puede llegar a ser más importante que aquellas que trabajan con la distancia recorrida.

En este contexto SimpliRoute se ve en la necesidad de abordar estas problemáticas, entre las que destaca la de generar rutas que tengan cierta noción de zonificación y balanceen las rutas por tiempo.

El VRP es un problema NP-Hard, con lo cual para poder resolver problemas que sean aplicables en la vida real la utilización de heurísticas ha sido necesaria. En particular, como se observa en Matl et al. [4] en problemas de ruteo de vehículos con condiciones de balanceo se suelen usar exclusivamente heurísticas. Al momento de considerar metodologías para resolver el problema existen heurísticas que siguen la estructura llamada *cluster first - route second* que consisten en primero realizar una clusterización y posteriormente rutear dentro de dichas clusterizaciones, en [6, 7] se presentan heurísticas clásicas que trabajan esta metodología.

En esta tesis se abordará realizar una heurística basada en árboles expansores de peso mínimo (Minimum Spanning Trees o MST) para realizar una clusterización de la instancia manteniendo en consideración el balanceo de tiempo. En este trabajo consideraremos la variante del VRPTW con cantidad de vehículos fijos. Lo desarrollado, al ser una clusterización, puede ser usado como la parte de clusterización dentro de cualquier metodología que utilice las heuristicas llamadas *cluster first - route second*.

Se analizarán los resultados de la clusterización contra instancias de benchmark de Gehring y Homberger [8], se estudiarán la complejidad de los algoritmos propuestos y se presentarán técnicas que permiten aumentar su velocidad reduciendo la cantidad de arcos permitidos, tanto entre nodos como entre clusters. Además, se analizarán teóricamente relaciones entre los MST y el grafo de los k vecinos más cercanos lo cual no solo permitirá reducir la complejidad del algoritmo propuesto, si no que aplicaciones en otras áreas (por ejemplo [9–11]).

Estructura de la tesis

En el capítulo 2 se hará una recapitulación del estado del arte tanto del VRP como de estructuras relevantes en teoría de grafos que serán utilizadas durante el trabajo.

En el siguiente capítulo se presentará la heurística con condiciones de balanceo y se analizarán en detalle el algoritmo presentado, su complejidad y técnicas para aumentar la velocidad.

Posteriormente, se analizarán criterios bajo los cuales se puede garantizar que se mantiene optimalidad del MST al eliminar aristas de un grafo completo, en particular se encontraron cotas ajustadas bajo las cuales, en un contexto determinista, se puede garantizar que el grafo, G_k, de los k vecinos más cercanos contiene al MST del grafo original. Además en un contexto probabilista se mostrarán valores para k tales que G_k contiene el MST asintóticamente. Además, se estudiará los impactos que tienen estos resultados sobre la heurística desarrollada y cómo este resultado puede ser beneficioso para otras áreas del conocimiento.

Por último, se mostraran los resultados de la optimización tras el uso de la heurística y se comparara con otras formas de solución, además se observara su efecto en la solución de ruteo y que efectos tiene el proceso de eliminar aristas sobre la velocidad del algoritmo.

Capítulo 2

Antecedentes

2.1. Antecedentes del VRP

El problema del VRP ha sido estudiado ampliamente en los últimos años, tanto en modelos que lo llevan a ser más cercano a la realidad como en técnicas de resolución. Una de las variantes mas conocidas corresponde a la de VRPTW, donde se agrega la restricción extra de consideración de ventanas horarias. Para más detalles de las variantes ver Braekers et al. [1].

Al momento de buscar formas de resolución del VRP existen distintas estrategias. No solo están los modelos y formulaciones exactas, si no que también se encuentran distintos tipos de heurísticas y metaheurísticas.

Entre las metahuerísicas se pueden mencionar algoritmos genéticos, colonia de hormigas (Ant Colony Optimization en ingles) o búsquedas tabú. Este tipo de metahueristicas, de manera general, buscan distintos tipos de soluciones, a veces de manera paralela como es con Ant Colony, mediante ciertos criterios probabilistas, de esta manera se busca explorar el espacio de soluciones intentando no caer en óptimos locales.

Entre las heurísticas se pueden definir dos tipos: construcción y mejoramiento. Las primeras buscan obtener una solución inicial al problema de manera constructiva, generalmente de manera greedy. Mientras que las de mejoramiento dada una solución inicial buscan mejorar una solución dada en una vecindad de ella, por esto se vuelve relevante la noción de vecindad se utiliza estas nociones se puede definir definir de varias maneras.

Entre las heurísticas encontramos *cluster first - route second* o *route first - cluster second*, que consisten en utilizar estrategias de clusterización en conjunto con el ruteo. Por otro lado encontramos heurísticas de mejoramiento intra tour e inter tour que suelen ser trabajados en la forma de búsquedas locales (local searchs en ingles), este tipo de búsquedas consisten en perturbar una solución actual e ir iterando hasta que no se logra encontrar una mejor solución y caer en un óptimo local.

Existen distintos tipo de las heurísticas de mejoramiento, entre ellas existen dos que serán importantes para este trabajo, *Relocate* y *Exchange*, la primera consiste en eliminar un nodo de una ruta e insertando en otra posición, ya sea de la misma ruta u otra, mientras que el

segundo consiste en mover un primer nodo de una ruta a otra y un segundo nodo de la última a la primera, es decir un intercambio de nodos entre rutas. Si se analiza toda la vecindad que este tipo de heurísticas generan se observa que tiene una complejidad de $O(n^2)$ sufren de un problema con el costo computacional a medida que aumentan los tamaños a trabajar, esto es un problema que también sera relevante, para esto una de las técnicas utilizadas se llamada búsqueda granular (granular search) que consiste en considerar solo conexiones que estén espacialmente relacionadas, por ejemplo solo aquellas que estén cercanas en distancia, y con esto limitar el espacio de búsqueda de manera considerable.

Estas clasificaciones y recopilaciones de los algoritmos fueron recuperadas de [12–14].

Dentro de las metodologías *cluster first - route second* han existido distintos approachs para incorporar la noción de ventanas horarias. Por ejemplo en [15, 16] se realizan modifiaciones de la distancia entre dos puntos donde se incorpora las ventanas horarias y utilizan esta nueva noción para sus operaciones.

Hay situaciones donde el balanceo de la carga de trabajo es importante por la aplicación que se esta modelando, por ejemplo los ingresos o costos de los vehículos pueden estar asociados a los clientes que visitan por lo que se desea mantener un balance entre vehículos. Al momento de decidir que balanceo utilizar es necesario tener en consideración la aplicación del problema y las características del mismo, por ejemplo en Sivaramkumar et al. [5] se hace una investigación donde se aprecia que para un VRPTW es más relevante el balanceo por tiempo total que el balanceo por distancia.

Al momento de querer obtener una solución balanceada existen dos factores que, a priori, deben ser relevantes *que se va a balancear* y *que función de balanceo utilizar*. En [3, 4] se realizan investigaciones al respecto, se concluye que dado el recurso a balancear el efecto de la función de balanceo es marginal y que balancear más de un tipo de recurso es incompatible, luego la elección de *que se va a balancear* es lo que se vuelve más relevante.

En el mismo estudio Matl et al. [4] se muestra cómo es que existen pocas consideraciones sobre balanceo en VRPTW y menos de balanceo de tiempo. Entre aquellas se pueden mencionar [5, 17, 18], todas ellas basadas en algún tipo de metaheurística.

En este contexto un approach de clustering que tengan en consideración ventanas de tiempo y balanceo de tiempo es valioso pues permite utilizarse como una primera etapa para una solución inicial del problema. Además, como veremos, el clustering propuesto incluso es capaz de ser usado en un contexto sin condiciones de balanceo lo que le entrega un valor extra al momento de querer aplicar estrategias de *cluster first - route second* para el VRPTW.

2.2. Antecedentes de grafos

Definición 2.1 *Dado un grafo, E el conjunto de aristas y una función de costo $c\colon E \to \mathbb{R}$, para $k \in \{1, ..., n-1\}$ se define el grafo de los k-vecinos más cercanos como el grafo $G_k := (V, E_k)$ con*

$$E_k = \{e = uv \in E \mid u \in NN_k(v) \vee v \in NN_k(u)\},$$

donde $NN_k(u)$ corresponde a los vértices que son $k-$vecinos más cercanos del punto u.

Si $k \leq 1$ definimos $G_k := G_1$, si $k \geq n-1$ $G_k := G_{n-1}$ y si $k \in [1, n-1]$ $G_k := G_{\lceil k \rceil}$.

En este trabajo se usara el grafo de los $k-$vecinos más cercanos (abreviado k-NNG por sus siglas en ingles) sobre el grafo completo de n vértices, además es importante notar que para una función arbitraria de pesos c es posible que exista ambigüedad sobre el conjunto $NN_k(u)$ esto se puede solucionar si, por ejemplo, se supone que c es inyectiva. En el afán de la generalidad se presenta la definición de *no ambigüedad de los $k-$vecinos más cercanos.*

Definición 2.2 *Dado un grafo no dirigido $G = (V, E)$ y una función de costo $c : E \to \mathbb{R}$ decimos c no es ambigua para los $k-$vecinos mas cercanos si es que para todo $u \in V$ el conjunto $NN_k(u)$ es único.*

El grafo de los vecinos más cercano tiene aplicaciones en distintas áreas como biotecnología, data mining, machine learning (por ejemplo [19–21]). A pesar de lo anterior son pocas las propiedades teóricas que se conocen en general para este tipo de grafos, ver Bose et al. [22] para encontrar ciertas propiedades sobre estructuras similares.

En este trabajo utilizaremos la versión no dirigida del grafo de los k vecinos pues nos sera de utilidad, a pesar de lo anterior en Bose et al. [22] se define el k-NNG como un grafo dirigido y si S corresponde al conjunto de puntos lo denotan k-NNG(S), en este trabajo se utiliza otra notación para evitar confusiones. En el contexto dirigido el siguiente resultado es interesante pues nos presentan una de las pocas propiedades conocidas sobre los k-NNG.

Proposición 2.1 *Para todo conjunto de puntos S*

$$\frac{kn}{2} \leq |E(k\text{-}NNG(S))| \leq kn - \binom{k+1}{2}$$

y las cotas son ajustadas.

Demostración. Ver Proposición 3.1 de Bose et al. [22]. □

Como ya se menciono, en esta proposición k-NNG(S) corresponde a la versión dirigida del grafo, a pesar de lo anterior en dicho trabajo se especifica que al momento de calcular $|E(k\text{-NNG}(S))|$ aquellas aristas que sean bidireccionales se cuenta una sola vez.

En nuestro caso trabajamos con la versión donde las direcciones son suprimidas, independiente de esto podemos recuperar el mismo resultado.

Proposición 2.2 *Dado $G = (V, E)$ un grafo simple y una función de costo $c : E \to \mathbb{R}$ se cumple que*

$$\frac{kn}{2} \leq |E_k| \leq kn - \binom{k+1}{2}$$

Demostración. Notar que la diferencia entre la versión dirigida y no dirigida, además de las direcciones de las aristas, es que en este ultimo las aristas bidireccionales solo se consideran

una vez.

Dado lo anterior para calcular $|E_k|$ basta contar todas las aristas de la versión dirigida con la precaución de aquellas que sean bidireccionales consíderalas una sola vez, es decir estamos utilizando la misma forma de contar que la utilizada en la Proposición 2.1, con lo cual ese resultado sigue siendo valido en nuestro contexto pues ambos cardinales serán iguales. □

Otra estructura interesante que se puede construir es la del MST, que corresponde a buscar un árbol expansor de peso mínimo. Dentro de las formas de resolver este problema existen tres algoritmos clásicos Kruskal, Prim y Bor$\mathring{u}$vka.

El algoritmo de Bor$\mathring{u}$vka no solo es importante pues es la base de algunos algoritmos modernos (por ejemplo ver Bernard [23]), si no que su relación con los grafos k-NNG nos entrega un gran insight sobre como estos se pueden relacionar con el problema del MST. El algoritmo consiste en calcular G_1, luego se calcula nuevamente pero sobre el grafo donde los puntos son las componentes conexas de G_1 (ver Nešetřil et al. [24] para más detalles), en particular se obtiene la proposición 2.3 que es interesante pues nos garantiza que al menos una parte de las aristas se obtienen con $1 - NNG$.

Proposición 2.3 *Dado un grafo no dirigido $G = (V, E)$ y una función de costo $c : E \to \mathbb{R}$ inyectiva entonces existe T MST tal que*

$$E_1 \subseteq T$$

Demostración. La demostración es una consecuencia directa del algoritmo de Bor$\mathring{u}$vka pues, como se menciona en Nešetřil et al. [24], el algoritmo comienza seleccionando la arista más cercana a cada uno de los vértices, es decir hemos seleccionado todas las aristas de E_1 y se concluye lo requerido. □

Como corolario importante se obtiene que si G_1 es conexo entonces G_1 es el MST del grafo, cabe destacar que la hipótesis de inyectividad no es una limitante del algoritmo, lo que si es necesario es que exista una forma consistente de elegir puntos cuando estos tengan el mismo peso, en este sentido la *no ambigüedad de los $k-$vecinos más cercanos* se vuelve relevante para tomar estas decisiones sin inconvenientes.

Es posible definir estas estructuras en un contexto probabilista, donde en vez de una función de pesos cada una de las aristas distribuye como una variable aleatoria. En este contexto se tienen varias propiedades sobre la esperanza del peso y la distribución de la arista más pesada del MST o propiedades sobre la conectividad de G_k (por ejemplo ver [25–27]). Vale la pena explicitar el resultado de Cooper y Frieze [27] que nos dice

Teorema 2.1 *Si es que los pesos distribuyen de manera uniforme $[0, 1]$ iid entonces se cumple que*

$$\lim_{n\to\infty} \mathbb{P}(G_k \text{ sea conexo}) = \begin{cases} 0 & k = 1 \\ \gamma & k = 2 \\ 1 & k \geq 3 \end{cases}$$

donde $0.99081 \leq \gamma \leq 0.99586$ y n la cantidad de vértices en el grafo.

En este contexto, ya sea determinista o probabilista, surge una pregunta natural ¿Cuál es el menor valor de k tal que el MST este contenido en G_k? Sabemos que siempre G_1 esta contenido en el MST, pero el teorema 2.1 nos dice que casi seguramente se cumple que el MST no esta contenido en G_1 y que existe una probabilidad no nula de que tampoco este en G_2.

Buscando resultados de literatura se puede observar que no existe ningún trabajo que responda esta incógnita, en este escrito se le dará una respuesta para un modelo determinista cualquiera y se encontraran condiciones para el modelo aleatorio bajo las cuales es posible garantizar asintóticamente estos resultados, dado el desconocimiento del comportamiento de estas estructuras se vuelven valioso por si mismos estos resultados, pero veremos cómo es posible utilizarlo para ciertas aplicaciones y en los algoritmos que se plantearan durante la tesis.

Capítulo 3

Clusterización para balanceo de tiempo

En este capítulo se mostrará y desarrollará la heurística propuesta para resolver una clusterización que tenga en consideración balanceo de tiempos para el VRP, se analizarán la velocidad teórica del algoritmo y se presentaran herramientas que permiten aumentarla.

3.1. Consideraciones del problema

Como ya se mencionó existen varias variantes del VRP, durante todo el capítulo trabajaremos con un VRPTW con ciertas condiciones a tener en mente.

Consideraremos que las instancias que se resuelven son factibles, que se tendrá acceso a la matriz de tiempos de viaje y que esta es simétrica. Se tomará una cantidad fija de vehículos y que estos son homogéneos, tanto en capacidad como en el tiempo disponible. Además, al momento de resolver la clusterización, consideraremos un vehículo por cluster.

Al momento de resolver el problema se trabajara con un enfoque *Cluster First - Route Second* que separaremos en tres etapas fundamentales: *Clusterización*, *Solución Inicial* y *Mejora Solución Inicial*.

La clusterización se realizará mediante una heurística que busque capturar ciertas nociones relevantes tanto al momento de hablar de clustering como de ruteo. Dicha heurística busca tener en consideración características tanto de distancia como de demanda, capacidad, tiempo en ruta y ventanas horarias. Dado lo anterior y la forma de resolución que se propondrá puede separarse en dos vertientes distintas: VRP con ventanas horarias (VRPTW) y VRP sin ventanas horarias (CVRP).

La solución inicial consiste en resolver cada cluster como un problema de VRP con un único vehículo, para esto se utilizara el algoritmo de resolución de VRP que SimpliRoute facilito.

Por último, tras combinar cada una de las soluciones individuales en una gran solución, utilizando heurísticas de mejoramiento que tengan consideraciones de balanceo y zonificación. Es importante al combinar las soluciones seguir con la noción de balanceo, de otra forma dicha métrica se verá mermada a costa de métricas clásicas de optimalidad como el

tiempo en ruta.

3.2. Algoritmo de Clusterización

Antes de preguntarse cuales son las métricas de zonificación y balanceo, hay que preguntarse de qué manera se va a realizar la clusterización, este proceso se pensó y trabajo como una metodología de clustering más general, que permite ser usada en contexto más generales.

Realizaremos ciertos supuestos que nos ayudarán al momento de desarrollar las ideas, primero consideraremos que tenemos clusters iniciales dados, más adelante presentaremos como pueden ser determinados y a cada cluster se le va a asignar un único vehículo. En lenguaje de VRP estamos pidiendo una asignación inicial de vehículo y puntos, sin necesidad de un ruteo.

Para mejorar dicha clusterización inicial utilizaremos un proceso de local search, de esta manera la clusterización puede ser manejada, en su manera más abstracta, como se presenta en Código 3.1.

Código 3.1: Clustering

```
Function performClustering(Clusters)
   Do
      changeFound = False
      For localSearch in listOfLocalSearch:
         apply localSearch to Clusters
         If localSearch found movement:
            changeFound = True
         End If
      End For
   While changeFound
End Function
```

No es nueva la idea de realizar búsquedas locales en clustering (ver [28–30]) la ventaja de utilizar este tipo de estructura es que nos permite emular la situación que ocurrirá al rutear utilizando una estrategia de búsquedas locales iteradas, esto nos permite llevar cierta noción de una solución de ruteo sin tener que rutear de manera completa.

Esta versión del clustering tiene dos consideraciones importantes. En primer lugar, requiere que se le entreguen una cluserización inicial. En segundo, solo va a terminar si es que no encuentra mejores soluciones, esto podría ser modificado si se desea aumentar la velocidad o evitar caer en óptimos locales. Por otro lado, hay que recalcar que dado que existe una cantidad finita de posibles combinaciones este algoritmo termina, esto es suponiendo que los cambios aceptados disminuyen de manera estricta la función objetivo.

Se utilizarán dos que son inspirados en las heurísticas de VRP: *Relocate* y *Exchange*, es decir se buscará poder mover un punto de un cluster a otro y la posibilidad de intercambiar dos puntos. Solo se utilizan estas dos versiones pues al estar trabajando con clustering otros tipos de movimientos que trabajen sobre el orden de la ruta (como k-Opt) no hacen sentido

para este contexto.

Antes de presentar los algoritmos es importante definir la notación. $Clusters$ corresponde a la lista de clusters, utilizaremos $|Clusters|$ como su cardinal, denotaremos por n a la cantidad de clientes, $Cluster_i$ corresponde a un cluster de la lista y x_i será algún punto de $Cluster_i$.

Por simplicidad al local search asociado a Relocate se le llamara *ClusteringRelocate* y al de Exchange *ClusteringExchange*, tras esto se presentan los algoritmos en Código 3.2 y 3.3.

Código 3.2: Clustering Relocate

```
Function ClusteringRelocate(Clusters, costOfAcceptance, feasibleChange)
    bestCostFound = ∞
    howToDoBestChange = None
    For Cluster_i in Clusters
        For Cluster_j in Clusters
            If Cluster_i != Cluster_j:
                For x_i in Cluster_i
                    If change is feasible acording to feasibleChange
                        Calculate how to do change.
                        Calculate costOfChange.
                        If costOfChange < bestCostFound
                            Update bestCostFound
                            Update howToDoBestChange
                        End If
                    End If
                End For
            End If
        End For
    End For

    If bestCostFound < costOfAcceptance
        Change Clusters using howToDoBestChange.
    End If
End Function
```

Código 3.3: Clustering Exchange

```
function ClusteringExchange(Clusters, costOfAcceptance, feasibleChange)
    bestCostFound = ∞
    howToDoBestChange = None
    For i = 1 to |Clusters|
        For j = i + 1 to |Clusters|
            For x_i in Cluster_i
                For x_j in Cluster_j
                    If change is feasible according to feasibleChange
                        Calculate how to do change.
                        Calculate costOfChange.
                        If costOfChange < bestCostFound
                            Update bestCostFound
                            Update howToDoBestChange
```

```
                    end If
                end If
            end For
        end For
    end For

    If bestCostFound < costOfAcceptance
        Change clusters using howToDoBestChange.
    end If
end function
```

Preguntar si un movimiento es factible es clave al momento de estar trabajando en el VRP, pues no podemos aceptar movimientos que sabemos va a romper una de las restricciones del problema. Dado que no estamos ruteando realizaremos una factibilidad aproximada y tendremos en consideración la capacidad de los vehículos, el tiempo utilizado y de ventanas horarias, esto se ve en detalle en la sección 3.2.2.

Es claro que un Exchange es simétrico, es decir el cambio entre (i, j) es el mismo que el (j, i), esto hace que se pueda ahorrar parte de la búsqueda, el análogo a solo usar el triángulo superior de una matriz cuadrada, esto permite hacer la implementación levemente más eficiente.

Al momento de hacer la clusterización existen dos parámetros que son relevantes. El primero es cuando aceptar el movimiento, en general se aceptara si es que el la función objetivo disminuye estrictamente. El segundo es que es necesario considerar la factibilidad del cambio.

3.2.1. Función Objetivo

Dado que queremos obtener soluciones que sean *zonificadas* y *balanceadas por tiempo*, es necesario determinar cómo se van a medir estas consideraciones, con el objetivo de determinar que movimiento es más conveniente.

Para poder balancear por tiempo es necesario estimar el tiempo de viaje de los vehículos, para esto se decide utilizar un MST. Es conocido de Christofides [31] que utilizando un MST y un matching es posible obtener una aproximación de $\frac{3}{2}$ (worst case) del TSP, teniendo esto en consideración para un CVRP, si nos hacemos previamente cargo de la distribución de capacidad, es posible tener una aproximación del viaje lo cual permite balancear. Si se desea atacar un VRPTW es más delicado, pero se pueden utilizar principios similares para abordar el problema; esto nos va a generar dos algoritmos (muy similares) que permiten atacar CVRP y VRPTW.

Para trabajar vamos a construir dos MST que mantendremos por cada cluster, uno será asociado a la zonificación y lo llamaremos *zoneMST* y el otro asociado al balanceo y lo llamaremos *balanceMST*. El *zoneMST* se considerará utilizando solo los clientes de cada partición mientras que el *balanceMST* se utilizaran los nodos y el depot asociado al problema.

La razón para considerar de manera distinta la zonificación y el balanceo es debido a que,

si bien el depot es importante al momento de rutear, hay que considerar que pueden existir clusters que estén muy alejados del depot, por lo que incorporar al depot sería realmente tener un outlier dentro de la clusetrización. Tambien se realizaron experimentos de probar utilizar el depot dentro del MST de zonificación y los resultados asociados a la clusterización empeoran considerablemente con esa consideración.

Dado que estamos utilizando un MST utilizaremos como peso de las aristas los valores de la matriz de tiempos de viaje, de esta manera si tenemos el cluster i-esimo y denotamos por T_i^{zone} al MST de zonificación asociado al cluster i-esimo y a $w_{T_i^{zone}}$ a su peso, utilizaremos la notación análoga para el MST de balanceo. Ahora definimos la métrica de zonificación como:

$$\sum_{i=1}^{|Clusters|} w_{T_i^{zone}}$$

En términos del problema de clusterizar puntos no es nueva la idea de utilizar MST, por ejemplo en [32] se presenta un clustering jerárquico basado en MST. La bondad de utilizar esta función para realizar un clustering del VRP es que estamos buscando puntos que están cercanos en *ruta*, que es justo lo que uno esperaría que fuera lo que ocurriera en una solución del problema de ruteo.

Ahora es necesario determinar que métrica de balanceo. Se decide utilizar la métrica del rango, es decir:

$$\max_{i=1,\ldots,|Clusters|} Time_i - \min_{i=1,\ldots,|Clusters|} Time_i$$

donde $Time_i$ es el tiempo estimado en recorrer al cluster $i-$esimo.

Para poder trabajar con el balanceo es necesario poder estimar el valor del tiempo en ruta de cada cluster (conjunto de puntos) para calcular $Time_i$ se utilizarán dos metodos, uno asociado a los CVRP y otro a los VRPTW, la razón de esto es pues si somos capaces de hacernos cargo de la parte de las capacidades mediante la clusterización el problema principal se reduce a buscar una solución a un VRP o a entender las ventanas horarias. Es importante mencionar que el *balanceMST* solo se utilizara cuando no hay ventanas horarias.

3.2.1.1. Aproximación Tiempo En Ruta CVRP

En el caso del CVRP dado que cada cluster lo construimos con 1 vehículo, en realidad estamos ante un TSP con lo cual podemos utilizar, como ya se mencionó, el resultado de Christofides [31], ahora lo ideal sería no utilizar el matching para no aumentar la complejidad con lo cual utilizaremos una regresión lineal con dos componentes: el peso del MST con tiempo de servicio y la distancia promedio al depot, para obtener los resultados. Cabe destacar que el peso del MST con tiempo de servicio corresponde al peso del MST + suma de los tiempos de servicio del cluster.

No es nueva la idea de utilizar una regresión para abordar aproximaciones de los largos en ruta para TSP, CVRP e incluso VRPTW (ver [33, 34]), ahora la bondad de utilizar esta regresión lineal es que no utiliza explícitamente condiciones geométricas lo cual nos permite

evaluar en tiempo constante (dado el peso del MST) y es dependiente del método de resolución, lo cual puede ser beneficioso pues nos permite desligarnos de la potencia del solver utilizado.

3.2.1.2. Aproximación Tiempo En Ruta VRPTW

Para un VRPTW no es posible aproximar de buena manera el tiempo en viaje de ruta con el MST sin tener consideraciones extras, por ejemplo en Figliozzi [34] se introducen métricas asociadas a las ventanas horarias para poder capturar situaciones como: *estos dos puntos no pueden estar juntos en la ruta por ventana horaria* o *si quiero tener estos puntos juntos en ruta tengo que generar tiempo de espera*. De esta manera es necesario tener en cuenta otras consideraciones, dado que estamos usando un MST en las nociones de clustering uno esperaría que en distancia sea relativamente óptima la combinación de puntos que se obtienen, luego podemos enfocarnos solamente en atacar el problema de aproximar el tiempo de viaje de un set de puntos.

Consideremos que $\{[e_i, l_i]\}_{i=1}^{n}$ son las ventanas horarias de los puntos, denotamos por s_i el tiempo de servicio de i, $d_{i,j}$ el tiempo de viaje entre i,j y supondremos que evaluar $d_{0,\cdot}$ o $d_{\cdot,0}$ nos dará el tiempo al depot. Luego vamos a definir un pre-orden para estos puntos, mirar 3.1 para el detalle, de esta manera es posible ordenar la lista de clientes mediante este pre-orden.

$$iRj \iff (e_i < e_j) \vee (e_i = e_j \wedge l_i < l_j) \tag{3.1}$$

Notar que lo presentado en (3.1) es simplemente ordenar primero los clientes que empiezan antes y en caso de empezar al mismo tiempo se ordenan los que terminan antes, de esta manera le damos cierta prioridad a los clientes que es necesario atender antes. Notar que iRj y jRi si y solo si tienen la misma ventana horaria, en tal caso para decidir cual poner primero escogeremos el que tenga menor índice entre i y j.

Supondremos que tenemos los clientes ordenados de esta manera, en caso de no estarlo los ordenamos y renombramos, con esto presentamos el Código 3.4 que nos permite aproximar el tiempo de viaje para rutear n clientes y determinar cierta noción de infactibilidad de considerar este conjunto de puntos, es claro que si la métrica de infactibilidad es 0 entonces todos los puntos son ruteables pero el caso contrario no es necesariamente verdad, luego deseamos un número pequeño pero no necesariamente nulo.

Código 3.4: Estimar Tiempo TW

```
Function CalcularTiempo(Clientes, availableTime)
    Clientes se ordenan de acuerdo con R

    unFeasible = 0

    arrivalTime = empty list
    feasibleClientes = empty list

    tentativeTime = max(d_{0, 1}, e_{1})
```

```
    If (tentativeTime <= l_{1})
        arrivalTime.add(tentativeTime)
        feasibleClientes.add(1)
    else
        unFeasible++
    end If

    For i=2:|Clientes|
        tentativeTime = max(d_{i-1, i} + s_{i-1} + arrivalTime.last, e_{i})
        If (tentativeTime <= l_{1})
            arrivalTime.add(tentativeTime)
            feasibleClientes.add(i)
        else
            unFeasible++
        end If
    End For

    totalTime = arrivalTime.pop # Obtener el último elemento y eliminarlo de la lista

    For index=size(feasibleClientes):1
        i = feasibleClientes.get(index)
        tentativeTime = d_{i, 0} + s_{i} + totalTime
        If (tentativeTime  <= availableTime)
            totalTime = tentativeTime
            break
        else
            unFeasible++
            totalTime = arrivalTime.pop
        End If

    End For

    return unFeasible, totalTime
End Function
```

Como veremos más adelante este algoritmo nos permite tener una buena aproximación del tiempo en ruta, mientras que en términos de complejidad es claro que post ordenamiento es $O(n)$ y ordenar los puntos, utilizando, TimSort es posible realizarlo en $O(nlog(n))$ (ver Auger et al. [35]) luego obtenemos una complejidad total de $O(nlog(n))$.

Dado todo lo explicado anteriormente la función objetivo consistirá en:

$$\sum_{i=1}^{k} w_{T_i^{zone}} + \max_{i=1,\ldots,k} Time_i - \min_{i=1,\ldots,k} Time_i$$

Donde $Time_i$ puede ser calculado de distintas maneras dependiendo de la característica de la instancia. Notar que la función objetivo está planteada de tal manera que tanto la clusterización como el balanceo tengan similar importancia, si se desea que una componente sea más relevante se tiene que hacer una ponderación adecuada.

3.2.2. Factibilidad De Los Movimientos

Notar que al estar trabajando con un VRP existen ciertos movimientos de puntos que serán infactibles para el problema, aun cuando sean óptimos para el clustering, de esta manera es necesario calcular si el movimiento es posible realizarlo en un contexto de VRP. Mostraremos la lógica de los factibles (que llamaremos feasibles por su traducción al ingles) que están asociados al VRPTW, pues cuando no hay ventanas horarias son los mismos con la salvedad que se ignora todo lo dedicado a ellas.

Código 3.5: Feasible Clustering Relocate

```
Function isFeasibleClusteringRelocate(x_i, Cluster_i, Cluster_j)
    If not isCapacityFeasibleClusteringRelocate(x_i, Cluster_i, Cluster_j)
        return False
    end If

    If not isTravelTimeFeasibleRelocate(x_i, Cluster_i, Cluster_j)
        return False
    end If

    If not isTimeWindowFeasibleRelocate(x_i, Cluster_i, Cluster_j)
        return False
    end If
    return True
End Function
```

Código 3.6: Feasible Clustering Exchange

```
Function isFeasibleClusteringExchange(x_i, x_j, Cluster_i, Cluster_j)
    If not isCapacityFeasibleClusteringExchange(x_i, x_j, Cluster_i, Cluster_j)
        return False
    end If

    If not isTravelTimeFeasibleExchange(x_i, x_j, Cluster_i, Cluster_j)
        return False
    end If

    If not isTimeWindowFeasibleExchange(x_i, x_j, Cluster_i, Cluster_j)
        return False
    end If
    return True
End Function
```

Como se puede apreciar Código 3.5 y 3.6 siguen la misma estructura que consiste en revisar si: (1) no sobrepaso la capacidad del vehículo (2) no sobre paso el tiempo disponible del vehículo (3) no aumento la cantidad de nodos infactibles por ventanas horarias. Cada una de dichas factibilidades puede apreciarse en el Anexo B, lo importante a destacar es que todas son calculables en $O(1)$; suponiendo que ya hemos calculado los tiempos de viaje del cambio que estamos probando.

3.2.3. MST dinámico y complejidad del algoritmo

Notar que cada vez que se desee evaluar la función objetivo este proceso tiene complejidad $\log(|Clusters|)$, eso pues para la parte de zonificación resulta en solo estudiar el cambio de los clusters modificados y el cambio en el rango puede ser calculado utilizando una min-max heap (ver Atkinson et al. [36]).

Por otro lado, como estamos utilizando MSTs para los calculos se vuelve necesario actualizarlos cada vez que se desea preguntar por un movimiento, como veremos mas adelante dicha actualización se puede hacer de varias maneras. Una primera forma de resolver esto es re construir los MSTs cada vez que se quiera consultar por la actualización y si consideramos los algoritmos clásicos para resolver el problema del MST tenemos que ellos son del orden de $O(m + n \log n)$, por simplicidad (pues en nuestro caso $m = n^2$) digamos que es $O(n^2)$, esta forma de atacar el problema es lo que llamaremos *método de clusterización base*.

Cabe mencionar, que como veremos, pueden existir leves diferencias en la complejidad dependiendo si estamos trabajando con el modelo CVRP o VRPTW.

Como primer resultado presentamos la complejidad del método de clusterización base en la Proposición 3.1.

Proposición 3.1 *Denotamos por T el número de iteraciones y si cada vez que se hace una consulta se re construye el MST. Obtenemos una complejidad de*

$$O\Big(Tn^4\Big)$$

para el CVRP y VRPTW.

Demostración. La demostración es tediosa, pero sencilla y se relega al Anexo A.1.

□

Como se puede apreciar la complejidad es cuartica en n, esto genera que a nivel practico se vuelva infactible realizar esta forma de clusterizar pues el aumento de tiempo de ejecución se vuelve considerable.

Para disminuir la complejidad se utilizará una estructura de MST dinámica, es decir buscaremos realizar actualizaciones del MST de manera eficiente. En Chin y Houck [37] se presentan dos algoritmos que permiten resolver las preguntas ¿Como agregar un punto a un MST? y ¿Como quitar un punto a un MST? sin tener que construirlo de cero. En particular se presenta un algoritmo que permite insertar un punto a un MST en $O(n)$ (ver Código 3.7) y un algoritmo que permite responder la pregunta para eliminar cualquier nodo en $O(n^2)$ (ver Código 3.8).

Código 3.7: Insert

```
1 # Input. Un MST $T = (V, E_T)$ en $G = (V, E)$ con función de costo $c(e)$, e \in E  y un nuevo v
    ↪ értice $z$ con costos en las aristas $(z, v)$ para $v \in V$. Notar que $T$ sera representado
    ↪ mediante una lista de adyacencia $L(v)$ para $v \in V$, y que $L(v)$ solo se usa para aristas
    ↪ en $T$.
```

```
# Output. El MST T' = (V', E'_T) para el nuevo grafo.
# Método. Inicialmente se marcan todos los vértices como nuevos, E'_T = ∅, y t una variable
    ↪ global. Se elige r ∈ V un vértice cualquiera y se ejecuta INSERT(r). Este
    ↪ procedimiento para cuando todos los verices son marcados como viejos.
    ↪ T' = (V', E'_T ∪ {t}) es el MST para este nuevo grafo.
# Comentario. t es una variable global y es la arista mas pesada en el camino entre w y z,
    ↪ mientras que m es la arista mas pesada entre r y z.
Function INSERT(r)
   r se marca como viejo.
   m = (r, z)

   For cada vértice w en L(r)
      If w marcado como nuevo
         Insert(w)
         k = la arista mas pesada entre t y (w, r)
         h = la arista mas liviana entre t y (w, r)
         E_{T}' = E_{T}' \cup \{h\}
         If c(k) < c(m)
            m = k
         End If
      End If
   End For
   t = m
End Function
```

Código 3.8: Eliminar Puntos

```
# Input. Un MST T representado por una lista de adyacencia L(v) para v ∈ V.
# Output. Un etiquetado l_v(r) para cada vértice interno r de T, donde se describe la
    ↪ estructura del árbol para unir las componentes conexas si es que el vértice r es
    ↪ eliminado.
# Método. Sea T' = (V', E'_T) un sub grafo conexo de T. T' indicara aquellos vértices
    ↪ marcados como 1. Inicialmente todos los vertices estan marcados como 0 y nuevos.
    ↪ Elegir cualquier vértice r ∈ V y marcarlo como 1 y viejo. Ejecutar LABELVERTEX(r)
    ↪ .
# Comentario. r es el nuevo vértice en T' y w va a ser el siguiente nuevo vértice.
Function LABELVERTEX(r)
   For cada vértice w en L(r)
      If w marcado como 0
         Marcar w como 1
         DIST(r, w)
         Marcar w como viejo
         LV(r, w, w)
         LABELVERTEX(w)
      End If
   End For
   t = m
End Function

# Comentario. LV actualiza los valores de l_v(r) en T' con w siendo el nuevo vertice en T', y
    ↪ llamando LV recursivamente mediante depth-first. G'' = (V'', E'') corresponde a
```

```
    ↪ considerar las componentes conexas como nodos.
Function LV(r, s, w)
   Marcar r como viejo
   flag = 0 # determina si es hoja
   For cada vértice v en L(r)
      If v marcado como 1 y nuevo
         flag = 1
         LV(v, r, w)
      End If
   End For

   If flag = 1
      If l_v(r) = ∅
         l_v(r) = (w, D(x)) con (w, r) ∉ T', x ≠ w # x es único.
      Else
         Usando T'' = (V'', E'') definido por l_v(r), y el nuevo vertice w con sus costos (desde
    ↪ el array D) para cada uno de los vertices en V'' como input, llamar INSERT(w).
    ↪ Usar el nuevo MST para actualizar el l_v(r).
      End If
   End If
End Function


# Input. Un MST T = (V, E_T) y T' = (V', E'_T) con V' ⊆ V y E'_T ⊆ E_T. T' es el MST en el
    ↪ grafo generado por V'. Todos los vertices de V' estan marcados como 1 y viejos
    ↪ excepto por v que esta marcado como 1 y nuevo, mientras que todos los vertices en
    ↪ V − V' estan marcados como 0.
# Output. El array D(r) con r ∈ V' − {v}, donde D(r) se define como sigue. Si para
    ↪ r ∈ V' − {v} se considera (r, s) la arista en el único camino entre r y v. Se define T'(r)
    ↪ como las componentes conexas de T' que contienen a r formada por eliminar a la
    ↪ arista (r, s) y sea D(r) ∈ V' el vértice tal que (D(r), v) sea la arista mas liviana entre
    ↪ T'(r) y v.
# Método. Encontar un r tal que (v, r) ∈ E'_T, es decir r incidente a v y que este marcado
    ↪ como 1 y viejo. Ejecutar DIST(r, v).
# Comentario. Dos tipos de marcas son necesarias. Todos los vértices en V' están marcados
    ↪ como 1 y todos los vertices no visitados están marcados como viejos.
Function DIST(r, v)
   Marcar r como nuevo
   For cada vértice w en L(r)
      If w marcado como 1 y viejo
         DIST(w, v)
         If c((v,m)) > c((v, D(w)))
            m = D(w)
         End If
      End If
   End For
   D(r) = m
End Function
```

Existen resultados más modernos que pueden reducir esta complejidad, por ejemplo en Nardelli et al. [38] se presenta como eliminar cualquier nodo posible en $O(m\alpha(n, m))$, donde

α es la función inversa de Ackerman. Mientras que en Gaibisso et al. [39], para grafos planares, se presenta un algoritmo lineal en m para resolverlo.

Estos resultados traen problemas al intentar implementarlo en nuestro contexto, por un lado en nuestro caso $m = n^2$ luego lo presentado en Nardelli et al. [38] o similares no es del todo útil pues atacan bien el caso de grafos poco densos. Por otro lado lo presentado en Gaibisso et al. [39] necesita que el grafo sea planar lo cual no se cumple por que el grafo es completo. Estos inconvenientes generan preguntas interesantes a nivel teórico de grafos, por ejemplo ¿Cuantas aristas es posible quitarle al grafo sin perder optimalidad? En cierto sentido queremos saber si se pueden eliminar suficientes aristas para poder ocupar alguno de los métodos, como veremos en el capítulo 4 la respuesta a esta pregunta es afirmativa.

Luego si utilizamos lo presentado en Chin y Houck y haciendo precálculos, es posible disminuirla complejidad de mayor manera a nivel teórico y sobre todo practico. Para esto lo que se propone es (1) Pre calcular como eliminar y agregar cada nodo. (2) Realizar las consultas de factibilidad y función objetivo. (3) Actualizar solo los clusters que fueron afectados.

Veremos que hacer este proceso disminuye la complejidad y, en la práctica, al actualizar solo donde se hacen modificaciones ayuda a poder ser más rápido. Dado que las factibilidades se hacen en tiempo constante el mayor gasto que nos falta considerar es: (1) Cálculo de las TW. (2) Cálculo de la función objetivo. Sabemos que el cálculo de las TW se puede hacer en $O(nlogn)$ mientras que la función objetivo se puede actualizar en $O(\log |Clusters|)$ y construirla en $O(|Clusters|)$, de ahora en adelante sera de utilidad denotar por $n_{clusters} := |Clusters|$ y $n_i := |Cluster_i|$. Esta discusión nos lleva a la siguiente proposición

Proposición 3.2 *Utilizando los algoritmos de [37] se obtienen las complejidades*

$$O\left(n^2 + (n + n_{clusters})^2 + T\left[n^2 \log(n_{clusters}) + (n + n_{clusters})^2\right]\right)$$

y

$$O\left(n^2 \log(n) + (n + n_{clusters})^2 + T\left[n^2 \log(n) + (n + n_{clusters})^2\right]\right)$$

para CVRP y VRPTW, respectivamente.

Demostración. Nuevamente los cálculos son sencillos pero largos y se pueden observar en Anexo A.2.

□

Luego si se sigue acotando los valores utilizando que $n_{clusters} \leq n$ se observar cómo es que la complejidad se logró reducir por un factor de $O(n^2/\log(n_{clusters}))$ o $O(n^2/\log(n))$ dependiendo del caso, lo cual es una rebaja considerable, pero sigue siendo costosa computacionalmente para instancias del orden de los 1000 puntos (llegando a demorarse horas).

Para mejorar la velocidad del algoritmo es posible introducir una noción similar a la de vecinos más cercanos, pero sobre los clusters. Como ya se menciono anteriormente existe una técnica para disminuir el tiempo de ejecución que consiste en eliminar a priori arcos muy largos y solo analizar dentro de un sub conjunto de posibilidades, esto nace de la intuición que uno espera que en buenas soluciones los puntos queden relativamente cercanos, además en [40] se presentan mas detalles donde verifican numéricamente que en las soluciones tam-

bién tiende a ser el caso. De esta manera podemos utilizar la misma idea, finalmente realizar un ruteo y uno espera que en clustering suceda un fenómeno similar. Para trabajar con esta noción necesitamos una definición:

Definición 3.1 *Dado C un conjunto de puntos, definimos los $k-$vecinos mas cercanos de C como sigue*

$$NN_K(C) := \bigcup_{u \in C} NN_K(u)$$

Esta noción, en cierto sentido nos permite decir que puntos están cerca de la nube de puntos que puede ser C. Notar que es posible que un punto de C forme parte de sus $k-$vecinos, esto puede escaparse de la intiuición con que se definen los vecinos, a pesar de eso no va a ser un inconveniente.

De esta manera somos capaces de limitar los movimientos entre clusters, si tenemos C_i, C_j dos clusters aceptaremos mover $x_i \in C_i$ a C_j si y solamente si $x_i \in NN_K(C_i)$, es decir solo moveremos el punto si es que esta en los vecinos del cluster al que queremos moverlo. De manera similar definimos el movimiento para Exchange y en este caso ambos puntos tienen que ser capaces de moverse.

Es importante notar que esta técnica no disminuye la complejidad en el worst case, a pesar de lo anterior es claro que realizar esto puede ayudar considerablemente en la rapidez del algoritmo a nivel practico pues nos permite estudiar muchas menos opciones.

Además, uno esperaría que desde un punto de vista de clusterización solo aquellos puntos que estén en los bordes del cluster se muevan y lo hagan a clusters cercanos lo que implica que una cantidad no menor de los vecinos ya van a pertenecer al cluster, disminuyendo aun más la cantidad a estudiar. Esto se puede ver ejemplificado en la Figura 3.1 donde se muestra en color azul aquellos puntos tales que sus 10 vecinos mas cercanos pertenecen al mismo cluster que ellos, es decir solo aquellos puntos en rojo formarían parte de la búsqueda local.

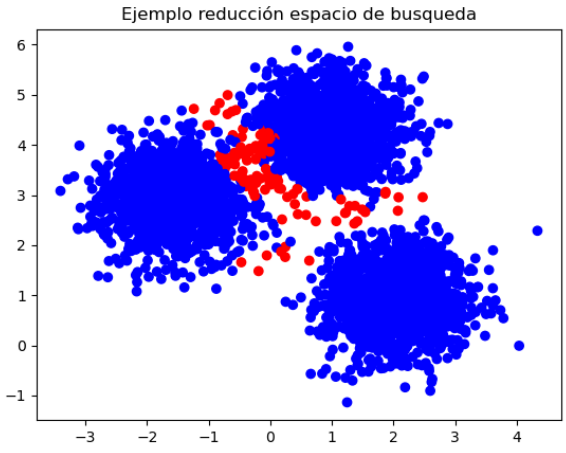


Figura 3.1: Reducción espacio de búsqueda para $k = 10$ vecinos

Es interesante notar como es que la restricción sobre los vecinos influye principalmente en la función objetivo del rango, esto es dado para la función de zonificación el óptimo del problema se encuentra en generar un MST del grafo completo y eliminar las $n_{clusters} - 1$ aristas mas pesadas que pertenecen al MST, ver proposición 3.3. Dado lo anterior, sabemos que se esperan conexiones no muy pesadas, es más si queremos asegurar un valor de K tal que no perdamos optimalidad en *zonificación* basta mirar cual es el valor de K tal que cumpla MST del grafo completo $\subseteq G_K$.

Proposición 3.3 *Suponiendo unicidad de los pesos entonces la función objetivo de zonificación alcanza su óptimo realizando la siguiente construcción: Se construye el MST del grafo completo y se eliminan las $n_{clusters} - 1$ aristas mas pesadas. Luego cada componente conexa forman la solución del problema.*

Demostración. Si T es un árbol denotaremos por $w(T)$ su peso y si e es arista lo escribiremos $w(e)$.

Denotemos por $\{T_i\}_{i=1}^{n_{clusters}}$ los arboles de la solución óptima y T al MST del grafo completo. Si consideramos los arboles como macro nodos y la arista entre dos arboles es representada como la arista mas pequeña entre ellos entonces es posible construir un MST con los macro nodos, es importante recalcar que a priori solo puede garantizar ser un árbol del grafo original.

Consideremos la siguiente separación de los arboles $(T_i, \{T_j\}_{j\neq i})$ es posible notar que estamos ante la presencia de un corte lo que implica que la arista que hace dicha conexión vive en el MST del grafo original. Esto lo podemos replicar para todo i, notar que es posible que existan repeticiones entre ellas. Luego se unen aquellas conexiones generando al menos un macro nodo menos, por tanto este proceso termina y es más termina cuando queda solo un macro nodo. Además notar que la arista que agrego genera que dos arboles se unan por tanto sigue siendo árbol, es decir al terminar este proceso vamos a tener un árbol y por tanto el proceso a lo mas puede durar $n_{clusters} - 1$ iteraciones.

Sea $\{\hat{e}_i\}_{i=1}^{n_{clusters}-1}$ entonces se tiene que $\sum_{i=1}^{n_{clusters}} T_i + \sum_{i=1}^{n_{clusters}-1} \hat{e}_i$ es un árbol, por tanto se cumple que

$$\sum_{i=1}^{n_{clusters}} w(T_i) + \sum_{i=1}^{n_{clusters}-1} w(\hat{e}_i) \geq w(T).$$

Sea $\{e_i\}_{i=1}^{n_{clusters}-1}$ las $n-1$ aristas mas pesadas del MST entonces se tiene que $T - \sum_{i=1}^{n_{clusters}-1} e_i$ es una solución factible del problema, pues cada vez que se elimina una arista aumenta en 1 la cantidad de componentes conexas. Esto implica que

$$w(T) - \sum_{i=1}^{n_{clusters}-1} w(e_i) \geq \sum_{i=1}^{n_{clusters}} w(T_i).$$

Con lo cual es posible obtener la siguiente desigualdad

$$\sum_{i=1}^{n_{clusters}} w(T_i) + \sum_{i=1}^{n_{clusters}-1} w(\hat{e}_i) - \sum_{i=1}^{n_{clusters}-1} w(e_i) \geq \sum_{i=1}^{n_{clusters}} w(T_i)$$

y asi se puede deducir

$$\sum_{i=1}^{n_{clusters}-1} w(\hat{e}_i) \geq \sum_{i=1}^{n_{clusters}-1} w(e_i).$$

Ahora, dada la construcción se cumple que $\hat{e}_i \in T$ para todo i y ahora si es que fueran las mismas aristas se tiene la igualdad directamente. Si no fueran podemos restar las que son iguales por lo que, sin perder generalidad, supondremos que son todas distintas. Luego por e_i ser las aristas mas pesadas del MST se tiene que $\hat{e}_j \leq e_i \ \forall i, j$ es decir las sumas son iguales por lo que se concluye que

$$\sum_{i=1}^{n_{clusters}} w(T_i) = w(T) - \sum_{i=1}^{n_{clusters}-1} w(e_i).$$

Es decir $T - \sum_{i=1}^{n_{clusters}-1} e_i$ es un conjunto de $n_{clusters}$ arboles y se obtiene el mismo peso que la solución óptima obtenida, por tanto la estrategia propuesta también entrega una solución óptima del problema. En particular también es posible deducir que el árbol generado a partir de T_i resulta ser MST del problema original.

□

Por otro lado, no es fácil calcular la reducción de velocidad, pues se vuelve necesario calcular

$$\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j \neq i}}^{n_{clusters}} \sum_{l=1}^{n_i - 1} n_j \mathbb{1}_{\text{el punto } l \text{ está en los } K \text{ vecinos más cercanos del cluster } j}$$

lo cual no es obvio, es posible notar (con cotas generosas) que existe una reducción clara cuando $K \leq n_{clusters}$. Esto se debe a que, por ejemplo, para calcular como agregar puntos hay que buscar todos los pares $(punto, clusters)$ o sea $n \cdot n_{clusters}$ casos y cada uno cuesta $O(n)$ con lo que hay $O(n^2 \cdot n_{clusters})$ y si utilizamos los K vecinos tenemos que cada punto se conecta a lo más con $\min\{n_{clusters}, K\}$ clusters por lo que se obtiene $O(n^2 \min\{n_{clusters}, K\})$, lo cual nos lleva a decir que claramente existe una disminución si es que K es pequeño en comparación a la cantidad de clusters. Algo importante a destacar es que en la práctica uno esperaría que la cantidad de conexiones a puntos fuera de clusters sea pequeña, lo razonable es que en general no existan muchas conexiones exteriores y las que existan solo se conecten a clusters realmente cercanos, básicamente apoyándose en la idea de que un punto de un cluster va a tener puntos cerca y por tanto el punto exterior está cerca de todos esos puntos.

3.2.4. Factibilizadores y el algoritmo completo

Al momento de trabajar con un proceso de clustering es posible que el clustering iniciales no sean lo suficiente buenos desde un punto de vista del VRP, por ejemplo pueden no ser factibles por carga o por ventanas horarias, luego para ayudar a la solución del problema se

genera un proceso que se llamara *Factibilizador*. De este modo existe un Factibilizador de Capacidad y un Factibilizador de Ventanas horarias.

Para el Factibilizador de Capacidad utilizaremos *ClusteringRelocate* y *ClusteringExchange* utilizaremos la misma función objetivo anterior con la diferencia que no nos importa empeorar la función objetivo en pos de beneficiar la factibilidad, y unos feasibles especiales que llamaremos *CapacityStrictFeasibles*, estos solo aceptaran movimientos si se mejora estrictamente la factibilidad de un cluster por capacidad, esto permite que los algoritmos terminen pues la condición de termino se asocia a si es posible mejorar estrictamente la infactibilidad (lo cual son una cantidad finita de casos), ver Anexo B.3 para los detalles de la implementación.

Para el Factibilizador de Ventanas horarias también utilizaremos $costOfAcceptance = \infty$ y unos feasibles especiales que llamaremos $TimeWindowStrictFeasibles$, que al igual que antes solo permiten movimiento si se mejora estrictamente la factibilidad de un cluster por ventana horaria independiente de si se empeora la función objetivo, ver Anexo B.4 para los detalles.

Esto resulta que realmente el proceso de clusterización sea 3 veces el proceso original con modificaciones leves de la aritmética que permite determinar movimientos factibles, por tanto, la complejidad no cambia.

Tras resolver el problema dentro de cada cluster se juntan todas las soluciones dentro de una gran solución y se sigue una búsqueda local análoga a la del clustering pero aplicada a la solución completa del VRP, esto es replicarlo lo hecho en la clusterización pero con rutas construidas en vez del MST pues nos inspiramos en heurísticas del VRP. Es importante mencionar que la función objetivo es necesario adaptarla para que tenga en consideración el balanceo, si no es posible que se pierda la noción que nos entrega el clustering, para esto utilizaremos la misma función objetivo, pero en vez de aplicada sobre los pesos del MST o la estimación de tiempo utilizaremos los verdaderos valores de las rutas.

Capítulo 4

Minimum Spanning Tree y k Nearest Neighbour Graph

Como se menciono anteriormente si se puede disminuir el espacio de busca en las búsquedas locales es posible aumentar la velocidad del algoritmo. Se espera que conexiones *muy grandes* no formen parte de la solución óptima, luego si se considera un espacio de búsqueda que solo contenga conexiones que sean *pequeñas* es posible disminuir el tiempo que se utiliza en estas búsquedas con leves efectos en la función objetivo. La idea anterior esta inspirada en la técnica de granular search para disminuir búsquedas cuadráticas en tabu search a nivel practico (ver Toth y Vigo [40]), en nuestro contexto nos encontraremos con un problema adicional de definir que significa que las conexiones sean *pequeñas* para el MST.

Otro beneficio de reducir la cantidad de aristas a considerar es que para eliminar puntos de un MST existen resultados modernos del problema ANR (All Node Replacement), el cual es clave para determinar cómo eliminar nodos de manera eficiente, entre ellos existe un algoritmo que es posible hacerlo en $O(m\alpha(n,m))$, luego si logramos disminuir la cantidad de aristas en un cluster tal que $m\alpha(n,m) \in o(n^2)$ entonces utilizar dicho algoritmo va a resultar ser más eficiente.

Para atacar lo mencionado estudiaremos propiedades sobre eliminación de aristas en grafos completos y como se relacionan con la optimalidad del MST. En primer lugar, veremos que hace sentido considerar la pregunta de eliminar aristas y mantener optimalidad del MST, luego veremos propiedades sobre el grafo k-NNG que nos garantiza propiedades deterministas o asintóticas en probabilidad sobre el MST y por ultimo veremos aplicaciones e intuiciones detrás de estos resultados, tanto en el algoritmo propuesto como en otras áreas del conocimiento.

Antes de continuar damos una definición que será de utilidad pues es tal que gira en torno a lo que deseamos probar en todo el capítulo.

Definición 4.1 *Sean $G = (V, E)$ y $G' = (V, E')$ dos grafos tales $E' \subseteq E$ y tanto G como G' son conexos. En tal caso decimos que G' es optimal si el MST de G' tiene el mismo peso que el MST de G.*

Otras nociones relevantes que nos ayudaran a trabajar en las demostraciones se presentan

a continuación. Siempre utilizaremos el grafo completo de n vértices, el cual lo denotamos por K_n. En primer lugar, introduciremos una noción de componentes conexas asociadas a eliminación de aristas en un árbol.

Definición 4.2 *Sea T un arbol de K_n, dado $e = uv$ definimos $V_{cc(u)}$ como los vértices en la componente conexa resultante de T al eliminar e y que contenga a u, de manera análoga lo hacemos para $V_{cc(v)}$. Si no hay ambigüedad escribiremos $cc(u)$ y $cc(v)$.*

Observación 1 Si es que eliminamos $e = uv$ de T se obtiene que $|cc(u)| + |cc(v)| = n$.

Si G' es un sub grafo conexo de G se tiene que el peso del MST de G' es mayor o igual al de G, por tanto, la noción de *optimal* nos dice cuando en verdad el MST estaba originalmente incluido en G'. De la definición 4.1 surgen dos preguntas naturales si existe un grafo $G' \neq G$ que cumpla lo deseado y que tanto es posible reducir el conjunto de aristas.

A priori no es claro que sea posible eliminar muchas aristas de un grafo y que el subgrafo sea optimal, de hecho, veremos que la forma de eliminar aristas resulta ser relevante. La Proposición 4.1 nos garantiza que preguntarnos eliminar aristas hace sentido.

Proposición 4.1 *Sea K_n el grafo completo de $n \geq 3$ vértices, $c : E \to \mathbb{R}$ una función de pesos tal que verifica*

$$c(e) < c(e_j),\ \forall e \in E \backslash \{e_i\}_{i=1}^{k},\ \forall j = 1, .., k$$

donde $k \in [1, ..., n-2]$ y $\{e_i\}_{i=1}^{k}$ son las k aristas más pesadas. Luego, dado T un MST se tiene que $e_i \notin T\ \forall i = 1, ..., k$.

Demostración. Supongamos que el resultado es falso, es decir $e_{\hat{i}} \in T$ para algún $\hat{i}$.

Ahora notamos que si $e \in T$ tenemos que si agregamos un ciclo y e forma parte de él entonces necesariamente esta arista es más liviana, o en el peor de los casos mismo peso que la agregada. Notar que se pueden armar $|cc(u)||cc(v)| - 1$ ciclos que utilizan a e pues la única forma de conectar entre elementos de $cc(u)$ y $cc(v)$ a través del árbol es por e, luego si verificamos que:

$$k - 1 < |cc(u)||cc(v)| - 1$$

llegaremos a una contradicción porque significa que podremos elegir $e \notin \{e_i\}_{i=1}^{k}$ tal que el ciclo que forma $T + e$ contiene a $e_{\hat{i}}$, esto implica que al considerar $T + e - e_{\hat{i}}$ obtendremos un árbol de menor peso, además utilizamos $k-1$ pues de las k aristas más pesadas ya usamos $e_{\hat{i}}$.

Notar que $|cc(u)||cc(v)|$ es equivalente a $|cc(u)|(n - |cc(u)|)$ y dado que sin perder generalidad podemos suponer que $|cc(u)| \leq |cc(v)|$ obtenemos que $|cc(u)| \leq \frac{n}{2}$, de esta manera queremos encontrar el menor valor posible de la función $f(x) = x(n - x)$ en el intervalo $[1, \frac{n}{2}]$. Dado que es una función cuadrática y el mayor valor se encuentra en $\frac{n}{2}$ obtenemos que $f(1) \leq f(x)$ con lo que $1(n-1) \leq |cc(u)||cc(v)|$ así obtenemos que

$$k - 1 \leq (n-2) - 1 < (n-1) - 1 \leq |cc(u)||cc(v)| - 1$$

demostrando lo pedido.

□

Notar la importancia de solo poder eliminar $n-2$ aristas, pues si eliminamos $n-1$ podríamos eventualmente eliminar todas las aristas de un punto y dejar disconexo el grafo.

Por otro lado, en cierto sentido esta propiedad puede verse como un punto intermedio entre que la función de pesos sea inyectiva y que puedan existir aristas con el mismo peso. Lo anterior nos permite garantizar que funciona independiente del MST, en la Proposición 4.2 se presenta una leve genralización independiente de la función de pesos.

Proposición 4.2 *Sea K_n con $n \geq 3$ vértices, $c : E \to \mathbb{R}$ una función de pesos y sean $\{e_i\}_{i=1}^{n-2}$ las $n-2$ aristas más pesadas. Entonces existe T MST tal que $e_i \notin T,\ \forall i = 1, .., n-2$.*

Demostración. Sea T un MST tal que contiene a $e_j \in T$ probaremos que es posible remplazarlo por $e \notin \{e_i\}_{i=1}^{n-2}$, de esta manera es posible eliminar las $n-2$ aristas de dicho árbol (de manera iterativa) y encontrar uno que cumple lo deseado.

Si escribimos $e_j = uv \in T$ entonces podemos replicar la demostración de la Proposición 4.1 en el sentido que sabemos que tiene que existir una arista e tal que tal que $e \notin \{e_i\}_{i=1}^{n-2}$, $e \notin T$ y $T + e$ forma un ciclo que contiene a e_j, con lo que $c(e_j) \leq c(e)$, luego por e_j ser parte de las aristas más pesadas siempre se cumple $c(e) \leq c(e_j)$ con lo cual $c(e) = c(e_j)$, notar que en este caso no hay contradicción pues no estamos suponiendo pesos distintos. Dado lo anterior podemos generar el árbol $T - e_j + e$ que es MST pues $c(T - e_j + e) = c(T)$ y $e_j \notin T - e_j + e$ demostramos lo deseado. □

Observación 2 Directamente se obtiene que es posible eliminar $k \leq n-2$ aristas.

La discusión anterior nos da visiones importantes sobre el problema. En primer lugar, es posible eliminar aristas y mantener la optimalidad del MST. En segundo lugar, la forma en que se eliminan aristas es relevante, pues en este punto intentamos eliminar las aristas más pesadas del todo el grafo, pero uno esperaría que argumentos similares por cada nodo funcionen, pues para cada nodo las aristas que forman parte del MST no pueden ser demasiado pesadas, veremos que el grafo k-NNG es una buena forma de representar esta idea.

Una intuición detrás de porque el grafo k-NNG es una buena forma de representar esta idea se encuentra en la propiedad de corte del MST. Para esto supongamos que la función de pesos es inyectiva, sea T el MST y $e = uv \in T$, luego tenemos que el conjunto $\{cc(u), cc(v)\}$ es una partición por tanto la propiedad de corte nos dice que la arista más liviana entre $cc(u)$ y $cc(v)$ está en el MST, en nuestro caso e. Lo anterior implica que hay $|cc(u)| \cdot |cc(v)| - 1$ aristas que son más pesadas que e, en particular e tiene que ser la más liviana de las $|cc(v)|$ conexiones que u puede hacer con el conjunto $cc(v)$, es decir, relativo a las conexiones de v (o de v al conjunto $cc(u)$) la arista no puede ser muy pesada, es más dado que sabemos que $|cc(u)|$ o $|cc(v)|$ es al menos $\frac{n}{2}$ la arista e es más liviana que $\frac{n}{2}$ de las conexiones de u o v,

dependiendo de cual de los dos conjuntos cumple la cota.

4.1. k Nearest Neighbour Graph

Durante esta sección abusaremos notación y diremos que $e = uv \in NN_k(u)$ si es que $v \in NN_k(u)$, es decir trabajaremos las aristas o los vértices de manera indistinta, a menos que genere ambigüedad.

Nos interesara mantener lo mas posible la generalidad así que siempre supondremos que se cumple la *no ambigüedad de los k−vecinos mas cercanos*, si es que no existe unicidad en general los resultados pueden ser remplazados de para todo MST a existe un MST.

El objetivo de esta sección será intentar responder la pregunta: ¿Que valor de k tiene que tomarse o bajo qué condiciones es posible asegurar que G_k es optimal?

4.1.1. Contexto determinista

Dado T un MST denotamos $\Delta(T)$ a su grado máximo, luego una primera intuición para el valor de k es decir que $k = \Delta(T)$, pues si queremos que $T \subseteq G_k$ entonces necesariamente $\Delta(T) \leq k$ pues es necesario que el nodo que tiene el mayor grado tenga al menos esa cantidad de vecinos. A pesar de esta idea lo anterior no siempre es verdad que se pude tomar $k = \Delta(T)$ pues el grafo puede quedar disconexo (ver Tabla 4.1) o si el grafo queda conexo puede no ser optimal (ver Tabla 4.2).

Tabla 4.1: Función de pesos tal que $G_{\Delta(T)}$ es disconexo

n	1	2	3	4	5	6
1		16	47	89	39	67
2	16		65	64	17	55
3	47	65		33	82	22
4	89	64	33		66	23
5	39	17	82	66		77
6	67	55	22	23	77	

Tabla 4.2: Función de pesos de $G_{\Delta(T)}$ con MST no optimal

n	1	2	3	4	5	6	7
1	0	625	821	458	480	799	478
2	625	0	442	645	282	884	300
3	821	442	0	254	430	159	522
4	458	645	254	0	418	380	679
5	480	282	430	418	0	613	386
6	799	884	159	380	613	0	402
7	478	300	522	679	386	402	0

En verdad el valor de $\Delta(T)$ no está relacionado con la posibilidad del valor de k, si no que la cota real (sin ningún supuesto) está relacionada a la cantidad de puntos presentes y no a otras estructuras. Esto se aprecia en el Teorema 4.1 donde se presenta una cota justa para el valor de k.

Teorema 4.1 *Sea $n \geq 3$, $k \in \{1, ..., n-1\}$. Bajo esas condiciones se cumple que:*

- *Si $k < \lfloor \frac{n}{2} \rfloor$ entonces existe una función de pesos $\hat{c}$ tal G_k no es optimal para $\hat{c}$.*
- *Si $\lfloor \frac{n}{2} \rfloor \leq k$ entonces para toda c que no genera ambigüedad en los $k-$vecinos mas cercanos se tiene que G_k es optimal.*

Demostración. Representaremos los puntos de K_n como $1, 2, 3, ..., n$. En primer lugar notar que basta demostrar el primer punto para $k = \lfloor \frac{n}{2} \rfloor - 1$ y el segundo para $k = \lfloor \frac{n}{2} \rfloor$ gracias a que $G_k \subseteq G_{k+1}$ en el sentido que si $e \in E_k \implies e \in E_{k+1}$.

Para $k = \lfloor \frac{n}{2} \rfloor - 1$ utilizaremos la siguiente función de costos

$$c(i,j) = \begin{cases} c(j,i) & j < i \\ i & j = i+1,\ i \leq \lfloor \frac{n}{2} \rfloor - 1 \\ j-2 & j = i+1,\ i \geq \lfloor \frac{n}{2} \rfloor + 1 \\ (n-2)+i & j = \lfloor \frac{n}{2} \rfloor,\ \ i \leq \lfloor \frac{n}{2} \rfloor - 2 \\ (n-2)+(j-2) & j \geq \lfloor \frac{n}{2} \rfloor + 2,\ i = \lfloor \frac{n}{2} \rfloor + 1 \\ 2(n-2)+1 & j = \lfloor \frac{n}{2} \rfloor + 1,\ i = \lfloor \frac{n}{2} \rfloor \\ +\infty & \sim \end{cases}$$

Donde en vez de $+\infty$ podemos usar pesos muy grandes y distintos, esto nos permite garantizar unicidad del MST. Sea T el único MST, luego lo único que nos basta demostrar es que una de las aristas de T no se encuentra en G_k.

Dada la construcción de la función de pesos $T = \{(i, i+1) \mid i = 1, ..., n-1\}$ es el MST. Esto se debe a que si $i \neq \lfloor \frac{n}{2} \rfloor$ la conexión $(i, i+1)$ es la mas liviana para i, mientras que la conexión $(\lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor + 1)$ forma parte del MST pues es la conexión mas liviana entre el corte dado por $\{1, 2, 3, ..., \lfloor \frac{n}{2} \rfloor\}$ y $\{\lfloor \frac{n}{2} \rfloor + 1, ..., n\}$.

Basta mostrar que la arista $(\lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor + 1)$ no puede estar en G_k para verificar lo deseado, esto se debe a la unicidad del MST y el hecho que G_k es subgrafo del grafo original.

Dada la construcción el conjunto $NN_k(\lfloor \frac{n}{2} \rfloor) = \{1, 2, 3, ..., \lfloor \frac{n}{2} \rfloor - 1\}$ mientras que $NN_k(\lfloor \frac{n}{2} \rfloor + 1) = \{\lfloor \frac{n}{2} \rfloor + 2, ..., n\}$. Esto implica que $(\lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor + 1) \notin N_k^E(\lfloor \frac{n}{2} \rfloor) \vee (\lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor + 1) \notin N_k^E(\lfloor \frac{n}{2} \rfloor + 1)$ con lo cual $(\lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor + 1) \notin G_k$ y concluimos lo deseado.

A continuación, probaremos el segundo punto para $k = \lfloor \frac{n}{2} \rfloor$, realizaremos el análisis lo mas general posible antes de ponerse en los casos de n.

Sea $e = uv \in T$, donde T es el MST, supongamos por contradicción que $e \notin NN_k(u) \wedge e \notin NN_k(v)$. Notamos que siempre podemos generar el siguiente tipo de ciclos $uvv_1v_2..v_lu$ donde v_i son elementos de $cc(v)$ cuando se elimina e de T (análogo para $cc(u)$) y gracias a la

propiedad de ciclos del árbol y que e pertenece a esos ciclos se tiene que es más liviano (o de igual peso) que todas las aristas de los ciclos. Luego si es que demostramos que podemos armar suficientes aristas es posible justificar (por palomar) que necesariamente una de las aristas de la forma (u, v_l) ó (u_j, v) es parte de las k mas livianas de u o v lo cual sería una contradicción a la no ambigüedad de c.

Para hacer lo anterior notamos que la cantidad de ciclos que se pueden armar, gracias a las propiedades de árbol, son $|cc(u)| - 1$ o $|cc(v)| - 1$, spg supondremos que $|cc(u)| \leq |cc(v)|$, luego sabemos que $\frac{n}{2} \leq |cc(v)|$. Ahora es claro que $|NN_k(u)^c| = |NN_k(v)^c| = n - 1 - k$ y dado que e no esta en ninguna de estos vecinos ya utilizamos una de estas aristas por tantos quedan $n - 2 - k$ disponibles, si demostramos que:

$$|NN_k(u)^c| = n - 2 - k < |cc(v)| - 1$$

por palomar necesariamente va a existir $v_l \in cc(v)$ tal que $v_l \in NN_k(u)$, de esta manera se tiene por un lado que $c(uv_l) < c(e)$ por la no ambigüedad mientras que por otra parte e es parte del ciclo generado por uv_l lo cual implica que $c(e) \leq c(uv_l)$ lo cual sería una contradicción a que T es MST y concluimos lo deseado.

Ahora nos ponemos en casos sobre n para demostrarlo, si n es par tenemos que

$$n - 2 - k = n - 2 - \frac{n}{2} = \frac{n}{2} - 2 < \frac{n}{2} - 1 \leq |cc(v)| - 1$$

con lo cual demostramos lo deseado. Ahora si n es impar es necesario notar que $\frac{n}{2}$ es un numero decimal y que $|cc(v)|$ es entero por tanto en verdad se cumple que $\frac{n+1}{2} \leq |cc(v)|$ con esto se tiene que:

$$n - 2 - k = n - 2 - \frac{n-1}{2} = \frac{n+1}{2} - 2 < \frac{n+1}{2} - 1 \leq |cc(v)| - 1$$

con lo cual se concluye lo deseado. □

Un corolario interesante de lo anterior es que nos garantiza que G_k es conexo si se toma un valor suficientemente grande de k.

Corolario 4.1 *Sea $k \geq \lfloor \frac{n}{2} \rfloor$ entonces G_k es conexo.*

Lo interesante del Corolario 4.1 es que mejora en cierto sentido el resultado presentado en el Teorema 2.1, pues no necesitamos suponer ninguna condición probabilista sobre la función de peso pero se pierde en el sentido que k depende de n. Por otro lado G_k optimal implica que G_k es conexo, por tanto uno espera que se necesite valores más grandes sobre k para ser optimal que para ser conexo.

Lo importante del Teorema 4.1 es que nos caracteriza cuando podemos hacer la eliminación, ya sabemos que no podemos hacerlo mejor que $\frac{n}{2}$ sin suponer algo más. Lo interesante de esto es que la estructura que lo hizo fallar es similar a la de clusrerización, es decir teníamos dos agrupaciones de puntos donde eran livianas entre ellas esto nos da a entender que si estuviéramos trabajo en estructuras clusterizadas un valor razonable a utilizar podría ser $\frac{n}{K}$ con K la cantidad de clusters.

Dado el MST es posible seguir mejorando la cota, por ejemplo de la demostración se deduce que

$$k = \max\{n - \max(|cc(u)|, |cc(v)|) \mid e = uv \in T\} = \max\{\min\{|cc(\hat{u})|, |cc(\hat{v})|\} \mid e = \hat{u}\hat{v} \in T\}$$

es un valor que cumple lo deseado y es una mejora a lo encontrado, sin embargo el problema de esto es que se vuelven dependiente del árbol o de suponer ciertas estructuras sobre la función de peso. En la siguiente sección atacaremos este segundo punto, y supondremos ciertas condiciones probabilistas.

4.1.2. Contexto probabilista

En toda esta sección trabajaremos con la versión probabilista de los grafos, es decir cada arista tiene un peso que distribuye como una variable aleatoria.

4.1.2.1. Resultados Teóricos

Antes de demostrar los resultados de esta sección mostraremos ciertos lemas que nos ayudaran en las demostraciones.

En primer lugar será de utilidad definir lo que es un estadístico de orden. Dado una muestra de n variables aleatorias el k-ésimo estadístico de orden corresponde al k valor mas pequeño de la muestra, para mas detalles ver David y Nagaraja [41].

Dado que una arista viva en G_k significa que el peso de esta es de las k aristas mas livianas de alguno de sus vértices es posible representar vivir en este grafo mediante su relación de los estadísticos de orden, considerando al conjunto de aristas que tienen vértices en común como muestras.

Lema 4.1 *Consideremos $X, \{X_i\}_{i=2}^{n-1}$ variables aleatorias continuas iid con cdf F_X. Dado $t \in \mathbb{R}$ y $k \in [1, ..., n-1]$ si denotamos por $X_{(k)}$ a su k-esimo estadístico de orden se cumple que*

$$\mathbb{P}(X = X_{(k)} \mid X = t) = \binom{n-2}{k-1} F_X(t)^{k-1}(1 - F_X(t))^{n-k-1}$$

Demostración. Notar que dado que $X = t$ lo que deseamos es que $k-1$ variables sean menores que t y el resto sea mayor, es decir tenemos que elegir $k-1$ de $n-2$ y no nos interesa en que orden por tanto se cumple que

$$\mathbb{P}(X = X_{(k)} \mid X = t) = \binom{n-2}{k-1} F_X(t)^{k-1}(1 - F_X(t))^{n-k-1}$$

□

A continuación, generalizaremos el resultado a 2 estadísticos de orden, lo cual nos va a permitir representar que una arista viva en los k-vecinos mas cercanos.

Lema 4.2 *Consideremos $X, \{X_i\}_{i=2}^{n-1}, \{Y_i\}_{i=2}^{n-1}$ variables aleatorias continuas iid con cdf F_X. Sea $t \in \mathbb{R}$ y $i, j \in [1, ..., n-1]$ si denotamos por $X_{(i)}$ al i-ésimo estadístico de orden de $X \cup \{X_i\}_{i=2}^{n-1}$ y de manera similar para $Y_{(j)}$ se cumple que*

$$\mathbb{P}(X = X_{(i)}, X = Y_{(j)} \mid X = t) = \binom{n-2}{i-1}\binom{n-2}{j-1} F_X(t)^{i+j-2}(1 - F_X(t))^{2n-i-j-2}$$

Demostración. Notando que dado X las variables $X_{(i)}, Y_{(j)}$ resultan ser independientes y utilizando el Lema 4.1 se concluye. □

El ultimo lema que presentaremos nos permitirá trabajar con la idea de *no pertenecer a los $k-$vecinos más cercanos*

Lema 4.3 *Bajo las mismas hipótesis que el Lema 4.2*

$$\mathbb{P}(X > X_{(k)}, X > Y_{(k)} \mid X = t) = f_{n,k}(t)$$

donde

$$f_{n,k}(t) = [\mathbb{P}(Z \geq k)]^2$$

con $Z \sim Binom(n-2, F_X(t))$

Demostración. Para esto notamos que:

$$\begin{aligned}
\mathbb{P}(X > X_{(k)}, X > Y_{(k)} \mid X = t) &= \Big[\mathbb{P}(X > X_{(k)} \mid X = t)\Big]^2 \\
&= \left[\sum_{i=k+1}^{n-1} \binom{n-2}{i-1} F_X(t)^{i-1}(1 - F_X(t))^{n-1-i}\right]^2 \\
&= \left[\sum_{i=k}^{n-2} \binom{n-2}{i} F_X(t)^{i}(1 - F_X(t))^{n-2-i}\right]^2 \\
&= [\mathbb{P}(Z \geq k)]^2 = f_{n,k}(t)
\end{aligned}$$

donde Z es como en el enunciado. □

Ahora estamos en condiciones de poder estudiar cómo se comportan las aristas de G_k, en particular mostraremos que bajos ciertas condiciones las aristas que no están en G_k no pueden ser muy pequeñas.

Proposición 4.3 *Consideremos que las aristas del grafo distribuyen como F_X iid variable aleatoria continua. Y sean $r, k \in \mathbb{R}$ dos numeros tal que*

$$F_X(r) < \frac{k}{n-2}$$

además suponemos que $k \in [1, ..., n-3]$. En tal caso, dado $e \in E$ se cumple que

$$\mathbb{P}(X_e \leq r, e \notin G_k) \leq \left[1 - \frac{k}{n-2}\right]^{-2(n-k-2)} \left[\frac{k}{n-2}\right]^{-2k} \left[1 - F_X(r)\right]^{2(n-k-2)} \left[F_X(r)\right]^{2k+1}$$

Demostración. Ahora, dado que $e = uv \notin G_k \iff e \notin NN_k(u) \wedge e \notin NN_k(v)$. Si escribimos por X_e la variable aleatoria del peso y representamos los $n-1$ vecinos de u como $X_e \cup \{X_i\}_{i=2}^{n-2}$ y a los vecinos de v como $X_e \cup \{Y_i\}_{i=2}^{n-2}$ tenemos que:

$$e \notin G_k \iff X > X_{(k)}, X > Y_{(k)}$$

con lo anterior tenemos que

$$\begin{aligned}
\mathbb{P}(e \notin G_k, X_e \leq r) &= \mathbb{P}(X_e \leq r, X_e > X_{(k)}, X_e > Y_{(k)}) \\
&= \int_{-\infty}^{\infty} \mathbb{P}(X_e \leq r, X_e > X_{(k)}, X_e > Y_{(k)} \mid X_e = t) f_X(t) dt \\
&= \int_{-\infty}^{r} \mathbb{P}(X_e > X_{(k)}, X_e > Y_{(k)} \mid X_e = t) f_X(t) dt \\
&= \int_{-\infty}^{r} f_{n,k}(t) f_X(t) dt
\end{aligned}$$

Notar que si logramos demostrar que

$$f_{n,k}(t) \leq \left[1 - \frac{k}{n-2}\right]^{-2(n-k-2)} \left[\frac{k}{n-2}\right]^{-2k} \left[1 - F_X(r)\right]^{2(n-k-2)} \left[F_X(r)\right]^{2k} \ \forall t \leq r$$

habremos demostrado lo pedido.

Para lo anterior utilizaremos la cota presentada en Arratia y Gordon [42] que dice que si Z es $Binomial(n, p)$ y si $a = \frac{k}{n}$ es tal que $p < a < 1$ entonces:

$$\mathbb{P}(Z \geq an) \leq e^{-nH(a,p)}$$

donde $H(a, p) = a \log\left(\frac{a}{p}\right) + (1-a) \log\left(\frac{1-a}{1-p}\right)$. En nuestro caso $Z \sim Binomial(n-2, F_X(t))$, $a := \frac{k}{n-2}$ de esta manera es claro que $a < 1$ y solo basta ver qué $F_X(t) < \frac{k}{n-2}$, ahora notando que $F_X(t) \leq F_X(r) < \frac{k}{n-2}$ entonces para todo $t \leq r$ es posible usar la cota.

$$
\begin{aligned}
f_{n,k}(t) &= \left[\mathbb{P}(Z \geq k)\right]^2 \\
&\leq e^{-2(n-2)H\left(\frac{k}{n-2}, F_x(t)\right)} \\
&= exp\left(-2(n-2)H\left(\frac{k}{n-2}, F_X(t)\right)\right) \\
&= exp\Big(-2(n-2)\Big[\frac{k}{n-2}\log\left(\frac{k}{n-2}\frac{1}{F_X(t)}\right) \\
&+ \left(1-\frac{k}{n-2}\right)\log\left(\left(1-\frac{k}{n-2}\right)\frac{1}{1-F_X(t)}\right)\Big]\Big) \\
&= exp\Big(-2k\log\left(\frac{k}{n-2}\frac{1}{F_X(t)}\right) - 2(n-k-2)\log\left(\left(1-\frac{k}{n-2}\right)\frac{1}{1-F_X(t)}\right)\Big) \\
&= \left[\frac{k}{n-2}\right]^{-2k}\left[\frac{1}{F_X(t)}\right]^{-2k}\left[1-\frac{k}{n-2}\right]^{-2(n-k-2)}\left[\frac{1}{1-F_X(t)}\right]^{-2(n-k-2)} \\
&= \left[1-\frac{k}{n-2}\right]^{-2(n-k-2)}\left[\frac{k}{n-2}\right]^{-2k}\left[1-F_X(t)\right]^{2(n-k-2)}\left[F_X(t)\right]^{2k}
\end{aligned}
$$

Con lo anterior nos basta entender el óptimo de la función

$$g(t) = \left[1-F_X(t)\right]^{2(n-k-2)}\left[F_X(t)\right]^{2k}$$

Dado que g es derivable estudiaremos su derivada:

$$
\begin{aligned}
g'(t) &= -2(n-k-2)\left[1-F_X(t)\right]^{2(n-k-2)-1}\left[F_X(t)\right]^{2k} f_X(t) \\
&+ 2k\left[1-F_X(t)\right]^{2(n-k-2)}\left[F_X(t)\right]^{2k-1} f_X(t) \\
&= \left[1-F_X(t)\right]^{2(n-k-2)-1}\left[F_X(t)\right]^{2k-1} f_X(t)\left[2k(1-F_X(t)) - 2(n-k-2)F_X(t)\right] \\
&= \left[1-F_X(t)\right]^{2(n-k-2)-1}\left[F_X(t)\right]^{2k-1} f_X(t)\left[2k - 2(n-2)F_X(t)\right] \\
&= 2\left[1-F_X(t)\right]^{2(n-k-2)-1}\left[F_X(t)\right]^{2k-1} f_X(t)\left[k - (n-2)F_X(t)\right]
\end{aligned}
$$

Ahora notar que si $t \leq r$ se tiene que

$$F_X(t) \leq F_X(r) < \frac{k}{n-2} \implies 0 < \left[k-(n-2)F_X(t)\right]$$

notando que $\left[1-F_X(t)\right]^{2(n-k-2)-1}\left[F_X(t)\right]^{2k-1} f_X(t) \geq 0$ se deduce que $g'(t) \geq 0$ en $[0,r]$ por tanto se cumple que

$$g(t) \leq g(r)$$

con lo cual se tiene que

$$
\begin{aligned}
f_{n,k}(t) &\leq \left[1 - \frac{k}{n-2}\right]^{-2(n-k-2)} \left[\frac{k}{n-2}\right]^{-2k} g(t) \\
&\leq \left[1 - \frac{k}{n-2}\right]^{-2(n-k-2)} \left[\frac{k}{n-2}\right]^{-2k} g(r) \\
&= \left[1 - \frac{k}{n-2}\right]^{-2(n-k-2)} \left[\frac{k}{n-2}\right]^{-2k} [1 - F_X(r)]^{2(n-k-2)} [F_X(r)]^{2k}
\end{aligned}
$$

□

Si bien este es un resultado intermedio que utilizaremos para las demostraciones de los resultados importantes por sí mismo es relevante pues nos asegura que las aristas que no viven en G_k no son muy livianas.

A continuación presentaremos uno de los resultados más relevantes, donde veremos que es posible garantizar que G_{k_n} contiene al árbol de manera asintótica para cierta sucesión k_n conveniente.

Teorema 4.2 *Sea un grafo donde el peso de las aristas distribuye iid como una variable aleatoria continua con cdf F_X y suponemos que*

- *Sea α_n una sucesión positiva y acotada superiormente tal que $\frac{1}{n^{\alpha_n}} \to 0$.*
- *F_X invertible.*
- *$X \geq 0$*

Dado

$$
k_n := (n-2)\left(r_n + F_X(r_n)\right) + \frac{(1+\alpha_n)\log(n) + \log(\log(n))}{2}
$$

donde

$$
r_n := F_X^{-1}\left(\frac{2(2+\alpha_n)\log(n)}{n}\right)
$$

entonces se cumple que

$$
\lim_{n\to\infty} \mathbb{P}(MST \subseteq G_{k_n}) = 1
$$

y lo hace tan rápido como $1 - \frac{2}{n^{\alpha_n}}$.

Demostración. Notar que si eventualmente $k_n \geq n-2$ entonces el resultado se tiene directamente, por tanto, sin perder generalidad podemos considerar $k_n \leq n-3$. Para $e \in E$ denotaremos por X_e su peso, notar que

$$
\mathbb{P}(MST \subseteq G_{k_n}) = 1 - \mathbb{P}(\exists e \in MST \backslash G_{k_n})
$$

así basta ver que $\mathbb{P}(\exists e \in MST \backslash G_{k_n})$ se va a cero, notar que

$$\mathbb{P}(\exists e \in MST \backslash G_{k_n}) = \mathbb{P}(\exists e \in MST \backslash G_{k_n}, X_e \geq r_n) + \mathbb{P}(\exists e \in MST \backslash G_{k_n}, X_e \leq r_n)$$

luego si vemos que cada una de las expresiones se va a cero se concluye lo deseado.

Antes de continuar presentamos un resultado de Steele [43] que dice que si F es una función de distribución y T el árbol entonces se cumple que

$$\mathbb{P}(\max_{e \in T} X_e > \lambda) \leq e n^2 e^{-nF(\lambda)/2}.$$

Dado lo anterior se cumple que

$$\mathbb{P}(\exists e \in MST \backslash G_{k_n}, X_e \geq r_n) \leq \mathbb{P}(\exists e \in MST, X_e \geq r_n) \leq \mathbb{P}(\max_{e \in MST} X_e \geq r_n) \leq e n^2 e^{-nF_X(r_n)/2}$$

y notando que

$$e n^2 e^{-nF_X(r_n)/2} = \frac{e}{n^{\alpha_n}}$$

se concluye que $\mathbb{P}(\exists e \in MST \backslash G_{k_n}, X_e \geq r_n) \to 0$. Por otro lado tenemos que

$$\begin{aligned}
\mathbb{P}(\exists e \in MST \backslash G_{k_n}, X_e \leq r_n) &= \mathbb{P}(\bigcup_{e \in E} e \in MST \backslash G_{k_n}, X_e \leq r_n) \\
&\leq \sum_{e \in E} \mathbb{P}(e \in MST \backslash G_{k_n}, X_e \leq r_n) \\
&\leq n^2 \mathbb{P}(e \in MST \backslash G_{k_n}, X_e \leq r_n)
\end{aligned}$$

Notamos que

$$(n-2)F_X(r_n) = (n-2)\frac{2(2+\alpha_n)\log(n)}{n} < 2(2+\alpha_n)\log(n) \leq k_n$$

concluyendo que es posible ocupar las cotas demostradas anteriormente, con lo que se tiene

$$\begin{aligned}
\mathbb{P}(e \in MST \backslash G_{k_n}, X_e \leq r_n) \leq &\left[1 - \frac{k_n}{n-2}\right]^{-2(n-k_n-2)} \left[\frac{k_n}{n-2}\right]^{-2k_n} \\
&[1 - F_X(r_n)]^{2(n-k_n-2)} [F_X(r_n)]^{2k_n} F_X(r_n)
\end{aligned}$$

Ahora por simplicidad escribimos $\beta_n := k_n - (n-2)F_X(r_n) = (n-2)r_n + \gamma_n$ donde $\gamma_n = \frac{(1+\alpha_n)\log(n)+\log(\log(n))}{2}$.

$$
\begin{aligned}
&\left[1-\frac{k_n}{n-2}\right]^{-2(n-k_n-2)}\left[\frac{k_n}{n-2}\right]^{-2k_n}\left[1-F_X(r_n)\right]^{2(n-k_n-2)}\left[F_X(r_n)\right]^{2k_n}F_X(r_n)\\
&\leq \left[\frac{k_n}{n-2}\right]^{-2k_n}\left[F_X(r_n)\right]^{2k_n}F_X(r_n)\\
&= \left[\frac{F_X(r_n)(n-2)}{k_n}\right]^{2k_n}F_X(r_n)\\
&= \left[\frac{k_n-\beta_n}{k_n}\right]^{2k_n}F_X(r_n)\\
&= \left[1-\frac{2\beta_n}{2k_n}\right]^{2k_n}F_X(r_n)\\
&\leq e^{-2\beta_n}F_X(r_n)\quad (*)\\
&= \frac{2(2+\alpha_n)\log(n)}{ne^{2(n-2)r_n}e^{\gamma_n}}\\
&\leq M\frac{\log(n)}{ne^{2(n-2)r_n}e^{2\gamma_n}}
\end{aligned}
$$

Importante notar que para (*) sea verdad es necesario que $-2\beta_n \geq -2k_n$ es decir $\beta_n \leq k_n$ lo cual se cumple gracias a la construcción e hipótesis.

Ahora notar que $\lim_{n\to\infty} r_n = F_X^{-1}(0)$ esto pues $2(2+\alpha_n)$ es acotado y es más lo hace decrecientemente por tanto:

$$(n-2)F_X^{-1}(0) \leq (n-2)r_n \implies \frac{1}{e^{(n-2)r_n}} \leq \frac{1}{e^{(n-2)F_X^{-1}(0)}}$$

esto implica que

$$M\frac{\log(n)}{ne^{2(n-2)r_n}e^{\gamma_n}} \leq M\frac{\log(n)}{n}\frac{1}{e^{2(n-2)F_X^{-1}(0)}}\frac{1}{e^{2\gamma_n}}$$

Ahora notar que $X \geq 0 \implies F_X(0) = 0$ por tanto $F_X^{-1}(0) = 0$ tenemos

$$\frac{\log(n)}{n}\frac{1}{e^{\gamma_n}} = \frac{\log(n)}{n}\frac{1}{\log(n)n^{1+\alpha_n}} = \frac{1}{n^{2+\alpha_n}}$$

con lo que

$$n^2\mathbb{P}(e \in MST\backslash G_{k_n}, X_e \leq r_n) \leq \frac{1}{n^{\alpha_n}} \to 0$$

es decir hemos demostrado que

$$\mathbb{P}(\exists e \in MST\backslash G_{k_n}, X_e \leq r_n) \to 0$$

así concluyendo lo deseado.

□

Observación 3 Es importante notar que cuando decimos F_X invertible nos referimos en el sentido tal que el dominio está restringido al soporte de X.

Lo relevante del Teorema 4.2 es que nos garantiza que asintóticamente y casi seguramente el grafo G_{k_n} contenga al MST. Dado esto si podemos asegurar que $|E_{k_n}| \in o(n^2)$ logramos garantizar que obtendremos una mejora en la complejidad de la heurística que planteamos anteriormente.

Si es posible realizar la reducción en ciertos casos, por ejemplo gracias al Corolario 4.2 podemos asegurar que $k_n \in O(\log(n)))$ para el caso donde los pesos son $U([0,1])$, por lo que se cumple que $k_n \in o(n^\alpha)$ con $\alpha > 0$ luego utilizando que $|E_{k_n}| \leq k_n n$ con lo que es posible asegurar $|E_{k_n}| \in o(n^2)$.

Corolario 4.2 *La distribución uniforme cumple el teorema anterior y es mas*

$$k_n \in O(\log(n))$$

Notar que esto es interesante pues nos garantiza que es posible obtener una mejora de la complejidad del algoritmo para ciertos casos del algoritmo. Ahora si consideramos el caso euclidiano, donde los puntos se posicionan en un cuadrado uniforme de largo 1 y la distancia entre los puntos es la distancia euclidiana presentamos el Teorema 4.3, el cual veremos es una generalización estricta a lo ya presentado.

Teorema 4.3 *Si estamos en el caso euclidiano entonces si*

$$k_n = (n-2)r_n + \frac{(1+\alpha_n)\log(n) + \log(\log(n))}{2}$$

donde

- α_n *es una sucesión acotada positiva tal que* $\frac{1}{n^{\alpha_n}} \to 0$.
- $r_n = \sqrt{\frac{\log(n)}{\pi n}}$

se tiene que

$$\lim_{n\to\infty} \mathbb{P}(MST \subseteq G_{k_n}) = 1$$

Demostración. Dado lo demostrado en [26], se tiene que $\mathbb{P}(\exists e \in MST \backslash G_1, X_e \geq r_n) \to 0$ por lo que la demostración de este teorema radica principalmente en calcular F_X y con eso ver qué $\mathbb{P}(\exists e \in MST \backslash G_{k_n}, X_e \leq r_n)$ se va a cero.

Consideremos los puntos (X_1, Y_1), (X_2, Y_2) en definimos

$$X := \sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2}$$

luego, gracias a que la distribución es uniforme *iid* entonces podemos pensar X esta dado por $\sqrt{A^2 + B^2}$ donde A, B son *iid* y distribuyen como $X_1 - X_2$, utilizando convolución obtenemos que:

$$f_A(t) = \begin{cases} 1+t & \text{si } -1 \leq t \leq 0 \\ 1-t & \text{si } 0 \leq t \leq 1 \end{cases}$$

así se tiene que:

$$F_A(t) = \begin{cases} \frac{t^2}{2} + t + \frac{1}{2} & \text{si } -1 \leq t \leq 0 \\ -\frac{t^2}{2} + t + \frac{1}{2} & \text{si } 0 \leq t \leq 1 \end{cases}$$

Ahora estamos en condiciones de calcular la distribución de A^2 tenemos que

$$F_{A^2}(t) = \mathbb{P}(A^2 \leq t) = \mathbb{P}(|A| \leq \sqrt{t}) = 2\sqrt{t} - t$$

con lo cual se tiene que:

$$f_{A^2}(t) = \frac{1}{\sqrt{t}} - 1$$

Ahora estamos en condiciones de poder calcular X, notar que la diferencia vive en $[-1, 1]$ por tanto el cuadrado de ella vive en $[0, 1]$ lo que implica que X vive en $[0, \sqrt{2}]$. Por simplicidad primero calcularemos X^2 tenemos que:

Utilizaremos que $t \leq 1$, realmente no nos interesa el otro caso.

$$\begin{aligned} F_{X^2}(t) &= \int_{-\infty}^{+\infty} F_{A^2}(t-x) f_{B^2}(x) dx \\ &= \int_0^1 F_{A^2}(t-x) \left[\frac{1}{\sqrt{x}} - 1\right] dx \\ &= \int_0^1 \mathbb{1}_{t-x\geq 0} \mathbb{1}_{t-x\leq 1} \left[2\sqrt{t-x} - (t-x)\right] \left[\frac{1}{\sqrt{x}} - 1\right] dx \\ &= \int_0^t \left[2\sqrt{t-x} - (t-x)\right] \left[\frac{1}{\sqrt{x}} - 1\right] dx \\ &= \frac{t^2}{2} - \frac{8}{3} t\sqrt{t} + \pi t \end{aligned}$$

Ahora estamos en condiciones de evaluar en X tenemos que:

$$F_X(t) = \mathbb{P}(X \leq t) = \mathbb{P}(X^2 \leq t^2) = \frac{t^4}{2} - \frac{8}{3} t^3 + \pi t^2$$

Ahora notar que si t es pequeño se tiene que $F_X(t) \leq t^2 \leq t$ de esta manera definiendo $\beta_n := \frac{(1+\alpha_n)\log(n) + \log(\log(n))}{2}$ obtenemos que

$$\begin{aligned}
n^2\mathbb{P}(e \in MST \backslash G_{k_n}, X_e \leq r_n) &\leq n^2 \left[\frac{F_X(r_n)(n-2)}{k_n}\right]^{2k_n} F_X(r_n) \\
&\leq n^2 \left[\frac{r_n(n-2)}{k_n}\right]^{2k_n} F_X(r_n) \\
&= n^2 \left[1 - \frac{2\beta_n}{2k_n}\right]^{2k_n} F_X(r_n) \\
&\leq n^2 e^{-2\beta_n} F_X(r_n) \\
&\leq M \frac{n^2}{e^{2\beta_n}} r_n^2 \\
&= \frac{M}{\pi} \frac{n \log(n)}{n^{1+\alpha_n} \log(n)} \\
&= \frac{M}{\pi} \frac{1}{n^{\alpha_n}}
\end{aligned}$$

concluyendo así lo deseado. □

El resultado es mas ajustado que el del teorema general pues para t suficientemente pequeño

$$F_X(t) \leq t^2$$

entonces $\sqrt{t} \leq F_X^{-1}(t)$. Así dado que $\frac{2(2+\alpha_n)\log(n)}{n} \to 0$ para n suficientemente grande

$$(n-2)\sqrt{\frac{\log(n)}{\pi n}} \leq (n-2)\sqrt{\frac{2(2+\alpha_n)\log(n)}{n}} \leq (n-2)F_X^{-1}\left(\frac{2(2+\alpha_n)\log(n)}{n}\right)$$

lo que implica que esto es una mejora sobre el teorema general, además es claro que $k_n \in O(\sqrt{n\log(n)})$ lo cual va a resultar ser importante.

Es relevante mencionar como estos resultados son sobre estimaciones del valor mas justo que es posible tomar, esto pues las cotas que se hicieron son burdas pues se ignora el hecho que $e \notin G_{k_n}$ o que $e \in MST$, dependiendo de cuál de las cotas, luego se espera que los valores límites de k_n sean más ajustados que los presentados.

La discusión anterior no solo revela la relación entre MST y G_{k_n}, si no que también nos da cierta noción de cómo se comporta el MST, el cálculo de $\mathbb{P}(e \in MST)$ es complejo y por eso intentamos no usarlo, de todas maneras cuando estamos en el caso $U([0,1])$ se vuelve relativamente claro que esta probabilidad no puede ser muy pequeña pues tenemos que:

$$\mathbb{P}(MST \subseteq G_1) = 1 - \mathbb{P}(\exists e \in MST \backslash G_1) \geq 1 - n^2\mathbb{P}(e \in MST \backslash G_1) \geq 1 - n^2\mathbb{P}(e \in MST)$$

y dado que G_1 es disconexo cuando $n \to \infty$ entonces $\lim_{n\to\infty} \mathbb{P}(MST \subseteq G_1) = 0$ con lo cual

uno espera que para n grande $\mathbb{P}(e \in MST) \geq \frac{1+\epsilon}{n^2}$ donde $\epsilon > 0$, de hecho podemos hacerlo mucho mejor si utilizamos la Proposición 4.4

Proposición 4.4 *Dado $k \in [1, ..., n-1]$ y si tenemos distribución $U([0,1])$ en los pesos se tiene que*

$$\mathbb{P}(e \in G_k) = \sum_{i=1}^{k} \left[2 \sum_{j=i+1}^{n-1} a_n(i,j) + a_n(i,i) \right]$$

donde $a_n(i,j) = \binom{n-2}{i-1}\binom{n-2}{j-1} B(i+j-1, 2n-i-j-1)$ y $B(x,y)$ es la función beta.

Demostración. Denotemos por k_u y k_v al valor de los $k-$vecinos que corresponde $e = uv$, es decir e es el k_u vecino más cercano de u y el k_v vecino más cercano de v, luego definimos $k_e = \min\{k_u, k_v\}$ esto implica que

$$\mathbb{P}(e \in G_k) = \sum_{i=1}^{k} \mathbb{P}(k_e = i)$$

con lo que nos basta ver que

$$\mathbb{P}(k_e = i) = 2 \sum_{j=i+1}^{n-1} a_n(i,j) + a_n(i,i)$$

Para esto notamos que k_e es separable en 3 eventos $k_u = i, k_u < k_v$, $k_v = i, k_v < k_u$, $i = k_u = k_v$ lo cual nos genera que

$$\mathbb{P}(k_e = i) = 2\mathbb{P}(k_u = i, k_u < k_v) + \mathbb{P}(i = k_u = k_v)$$

donde hemos juntados los primeros dos eventos pues en probabilidad son iguales, además es importante notar que dado que $k_\cdot \leq n-1$ al utilizar $i = n-1$ en único evento factible es $i = k_u = k_v$ por tanto $\mathbb{P}(k_u = n-1, k_u < k_v) = 0$ calcularemos la probabilidad pensando en $i < n-1$, pero a posteriori observaremos que lo hecho sigue siendo válido incluso para dicho caso.

Notamos que el evento $k_u = i, k_u < k_v$ es lo mismo que decir que $X_e = X_{(i)}, Y_{(i)} < X_e$, pues sabemos que $Y_{(i)} \neq X_e$ pero dado que $i < k_v$ necesariamente el estadístico al cual corresponde Y tiene que ser más grande que i, con lo cual se tiene que:

$$\begin{aligned}
\mathbb{P}(X_e = X_{(i)}, Y_{(i)} < X_e) &= \sum_{j=i+1}^{n-1} \mathbb{P}(X_e = X_{(i)}, X_e = Y_{(j)}) \\
&= \sum_{j=i+1}^{n-1} \int_0^1 \mathbb{P}(X_e = X_{(i)}, X_e = Y_{(j)} \mid X = t) \\
&= \sum_{j=i+1}^{n-1} \binom{n-2}{i-1}\binom{n-2}{j-1} \int_0^1 t^{i+j-2}(1-t)^{2n-i-j-2} \quad (*) \\
&= \sum_{j=i+1}^{n-1} \binom{n-2}{i-1}\binom{n-2}{j-1} B(i+j-1, 2n-i-j-1) \\
&= \sum_{j=i+1}^{n-1} a_n(i,j)
\end{aligned}$$

donde en $(*)$ usamos el Lema 4.2, lo anterior implica que

$$\mathbb{P}(k_e = i) = 2\sum_{j=i+1}^{n-1} a_n(i,j) + \mathbb{P}(i = k_u = k_v)$$

de la misma manera que antes obtenemos que $i = k_x = k_y$ es el evento $X = X_{(i)}, X = Y_{(i)}$ lo cual por definición es $a_n(i,i)$ concluyendo lo pedido.

Por ultimo notamos que $\sum_{j=i+1}^{n-1}$ solo hace sentido cuando $i < n-1$ y utilizando la convención usual de sumas se obtiene que cuando $i = n-1$ es cero, por tanto recuperamos el único conjunto factible y comprobando que la formula funciona para todo i. □

Lo anterior implica el Corolario 4.3, que se obtiene simplemente evaluando

Corolario 4.3 *Bajo las hipótesis de la Proposición 4.4 se tiene que*

$$\mathbb{P}(e \in G_1) = \frac{3n-5}{2n^2-5n+3}$$

luego, sabemos que $G_1 \subseteq MST$ con lo cual $\mathbb{P}(e \in G_1) \leq \mathbb{P}(e \in MST)$ esto implica que

$$\frac{3n-5}{2n^2-5n+3} \leq \mathbb{P}(e \in MST)$$

lo cual es una mejor cota a la ya encontrada, en efecto tenemos que $\mathbb{P}(e \in MST)$ esta acotada inferiormente por algo que se parece a $\frac{1}{n}$ en vez de $\frac{1}{n^2}$, esto es interesante pues nos ayuda a entender de mejor maneara cómo se comporta el MST, de hecho sabemos que en general

$$a_n(i,j) = \binom{n-2}{i-1}\binom{n-2}{j-1} \int_{t\in\mathbb{R}} F_X(t)^{i+j-2}(1-F_X(t))^{2n-i-j-2} f_X(t)dt$$

lo cual implica que

$$2\sum_{j=2}^{n-1} a_n(1,j) + a_n(1,1) \leq \mathbb{P}(e \in MST)$$

esto es extremadamente interesante pues $a_n(1,1) = \int_{t\in\mathbb{R}}(1-F_X(t))^{2n-4} f_X(t)dt$, luego si realizamos el cambio de variable $u = F_X(t)$ entonces

$$a_n = \int_0^1 (1-u)^{2n-4} du = \frac{1}{2n-3}$$

lo que implica que independiente de la distribución

$$\frac{1}{2n-3} \leq \mathbb{P}(e \in MST)$$

4.1.2.2. Simulaciones

Perdida de optimalidad

Como ya se demostró, bajo las hipótesis del Teorema 4.2, es posible garantizar que para n suficientemente grande G_{k_n} contiene al MST de G con alta probabilidad. Una pregunta natural que surge es que tanta optimalidad se pierde con G_k si se eligen valores de $k < k_n$. A nivel practico pareciera ser que, en ciertos casos, se necesitan valores comparativamente menores a los asintóticos encontrados, sobre todo para situaciones practicas donde n resulta ser acotado.

Por ejemplo, si tomamos los pesos $U([0,1])$ utilizaremos $k_n = 3$, $k_n = 4$ y $k_n = 5$ en la Figura 4.1 se puede apreciar dos simulaciones sobre la perdida de optimalidad en función del valor de k, en dicha figura graficamos el valor del MST de G_k dividió en el valor del MST real, de esta forma 1 corresponde a que se alcanzó el mismo valor óptimo y en caso de que el grafo G_k quede disconexo se utilizara el valor 2 como representación.

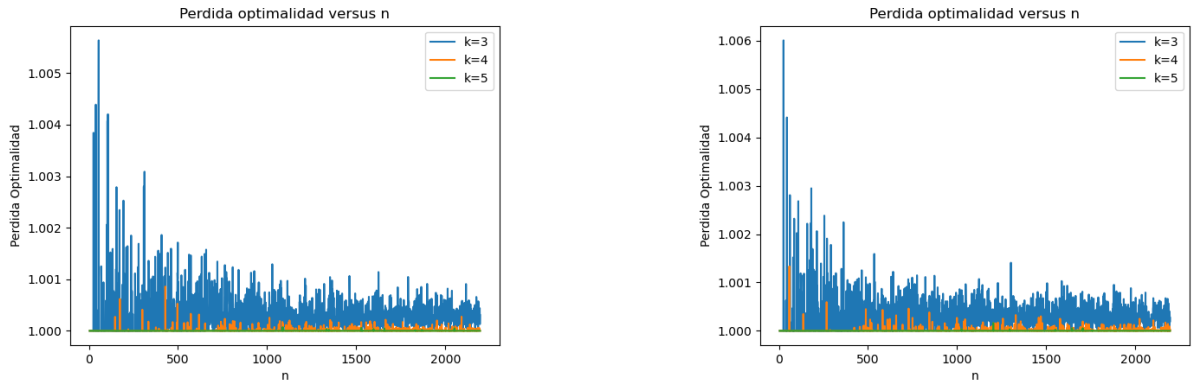


Figura 4.1: Simulaciones perdida de optimalidad

Se puede apreciar como la perdida de optimalidad es casi nula, incluso teniendo en consideración que estamos tomando valores de k fijos, y que al aumentar n pareciera tender a ser mas parecido el valor real con el utilizado. Esto da a entender, no solo que los resultados encontrados pueden ser considerablemente mayores al menor posible, también nos dice que la perdida de la función objetivo al considerar el MST restringido es pequeña, en las simu-

laciones no alcanza a superar el 0.6 %. Lo anterior da a entender que sería posible estudiar que tanta optimalidad se pierde al considerar G_k para un n fijo y que uno puede esperar que esta no sea alta.

Otra forma de notar que los valores encontrados pueden ser mayores a no ser ajustados, se encuentra en las demostraciones. En ellas se utilizo la siguiente cota

$$P(\bigcup_{e\in E} e \in MST\backslash G_{k_n}, X_e \leq r_n) \leq n^2\mathbb{P}(e \in MST\backslash G_{k_n}, X_e \leq r_n)$$

a prior esta cota pareciera que se está siendo demasiada burda pues sabemos que a lo más $n-1$ elementos por lo que se puede llegar a pensar que es posible acotar por un valor menor. En general uno esperaría que se puede utilizar $n\mathbb{P}(e \in MST\backslash G_{k_n}, X_e \leq r_n)$, a pesar de esto eso no es posible hacerlo pues al momento de considerar los $n-1$ estamos imponiendo la condición de que $e \in T$ por tanto es necesario condicionar para obtener la información extra. Es más, se puede observar cómo en simulaciones que considerar n en vez de n^2 es un valor mucho menor que la verdadera probabilidad, para ver esto y por simplicidad consideraremos $r_n = +\infty$ y $k_n = 3$.

Las simulaciones las realizaremos usando Monte Carlo, dado que cada probabilidad la podemos interpretar como la esperanza de una indicatriz, esto nos genera la Figura 4.2 donde la curva naranja corresponde a $n^2\mathbb{P}(e \in MST\backslash G_3)$, la verde a utilizar n y la azul la verdadera probabilidad. Como se puede apreciar la cota con n es considerablemente menor al valor de la probabilidad real, incluso para valores pequeños de n, mientras que la cota con n^2 si se encuentra como cota superior de la verdadera probabilidad. Además, es posible notar como la probabilidad con n^2 es una sobre estimación del valor deseado lo cual vuelve a reforzar la idea de que se están trabajando con cotas burdas y por tanto los resultados no necesariamente van a ser lo más ajustados posibles.

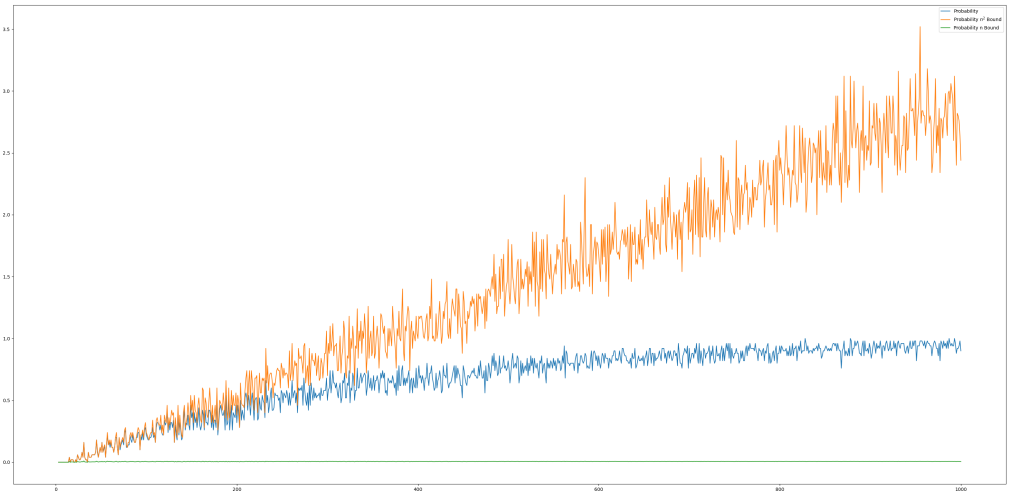


Figura 4.2: Probabilidad y su comparación con las cotas

4.2. Aplicaciones

Dado que ahora somos capaces de reducir la cantidad de aristas obtenemos la Proposición 4.5 que nos dice que el algoritmo de clusterización puede hacerse mas rápido, al menos dentro de sus sub proceso.

Proposición 4.5 *Si estamos bajos las hipótesis del Teorema 4.2 o 4.3 es posible resolver el problema de eliminar todos los puntos de un MST en $O(|E_{k_n}|\alpha(n, |E_{k_n|}))$ con alta probabilidad.*

Lo interesante de lo anterior es que $|E_{k_n}| \leq nk_n$, por tanto (ignorado la función inversa) se obtiene que es similar a $O(k_n n)$ así si es que $k_n n \in o(n^2)$ se genera una mejora en velocidad cada vez que se desee eliminar puntos.

Si estamos en el caso euclidiano, una hipótesis razonable en el contexto del VRP, tenemos que $k_n n \in O(\sqrt{n^3 \log(n)})$ por tanto logramos reducir la cantidad de iteraciones asociadas a la eliminación de puntos, lo anterior nos garantiza que somos capaces de aumentar la velocidad y mantener la optimalidad. A pesar de lo anterior, la complejidad del algoritmo sigue siendo similar y disminuye levemente, lo que realmente se logra hacer es disminuir la cantidad de operaciones n^2 que hacer, esto se aprecia en la Proposición 4.6.

Proposición 4.6 *Utilizando los algoritmos propuestos anteriormente, utilizando [38] y el grafo G_k para el caso euclidiano, si se define $h = \max_{i\in[3,\dots,n], j\in[1,\dots,i(i-1)/2]} \alpha(i,j)$ entonces para una cantidad de puntos suficientemente grande y tal que en cada cluster se tengan suficientes puntos se tiene que*

$$O\Big(n^2 + h\sqrt{(n+n_{clusters})^3 \log(n+n_{clusters})} + T\left[n^2 \log(n_{clusters}) + h\sqrt{(n+n_{clusters})^3 \log(n+n_{clusters})}\right]\Big)$$

y

$$O\Big(n^2 \log(n) + h\sqrt{(n+n_{clusters})^3 \log(n+n_{clusters})} + T\left[n^2 \log(n) + h\sqrt{(n+n_{clusters})^3 \log(n+n_{clusters})}\right]\Big)$$

para el CVRP y VRPTW respectivamente.

Demostración. El cálculo de las complejidades es similar a los anteriores y se relega al Anexo C.1. □

Notar que esto si es una reducción a lo presentado en la Proposición 3.2, esto debido a que en [44] se menciona que $\alpha(n, m)$ alcanza su máximo cuando $m = n$ con lo que se tiene que $h = \max_{i=1,\dots,n} \alpha(i, i)$ y luego observando la definición de α en [44] tenemos que $h = \alpha(n, n)$ y dado que $\alpha(n, m)$ crece extremadamente lento se tiene que $\alpha(n, n) \in o\left(\frac{n}{\log(n)}\right)$, esto se deduce del hecho que la función de Ackerman crece más rápido que una exponencial (ver la Sección

2.7 de [45]).

La importancia de este análisis no solo radica en el hecho de que es posible considerar muchas menos conexiones de las que aparentemente son necesarias, si no que nos da ciertas luces de los resultados vistos en los K vecinos por cluster, esto debido a que en la estructura de los MST se cumple y, por tanto, de cierta manera se puede extender al clustering.

Como ya se mencionó al tener un MST en cada cluster tenemos que el MST de la totalidad del grafo está dada por re conectar cada uno de estos clusters como macro nodos, ahora sabemos que estas reconecciones no son más pesadas que los $\frac{n}{2}$ vecinos mas cercanos o k_n en general, si tomamos el approach probabilista. Esto es relevante pues dada la discusión de optimalidad de la función objetivo de zonificación nos entrega un primer acercamiento a que es posible aumentar la velocidad sin perjudicar esa parte de la optimalidad, lo que deja abierto a seguir entendiendo cómo se comporta el movimiento para la función en general.

Por otro lado, tal y como se menciona en [11], existen estudios en Física, Cosmología y Astrofísica donde utilizan MST, es mas ahí mismo utilizan un grafo de KNN para poder realizar los cálculos del MST. Tras lo demostrado si es que hacemos supuestos probabilistas o tomamos $k = n/2$ es posible ajustar el valor de k tal que aumentemos la velocidad de cálculo y manteniendo la optimalidad del MST, ya sea siempre o con harta probabilidad.

Por ultimo cabe destacar que este análisis permite justificar que la idea utilizadas en granular search son validas en el contexto del MST, es decir es posible reducir el espacio de búsqueda sin perder optimalidad. Esto da a entender que sería posible ahondar más en estas técnicas y buscar obtener garantías sobre problemas mas complejos como lo son el TSP y VRP, lamentablemente las demostraciones se basaron fuertemente en propiedades y resultados del MST, por tanto no es directo poder extender estos resultados a dichos problemas, pero si dan una intuición de que puede ser beneficioso hacerlo. Además es posible observar que para el TSP es posible decir que el algoritmo de Christofides ([31]) puede ser ajustado haciendo esta limpieza de aristas al momento de la construcción del MST, luego sabemos que antes de realizar el matching si es posible garantizar que hay varias aristas que a priori pueden ser ignoradas, bajo esto solo faltaría entender como el matching se vería afectado con esta reducción y si es posible mantener las garantías del algoritmo con esta eliminación o si al momento de realizar esa parte es necesario volver a utilizar todas las aristas o otra cantidad disponible.

Capítulo 5

Resultados numéricos

5.1. Instancias Utilizadas

En este capítulo se utilizaran dos tipos de instancias, aquellas con las que probaremos lo realizado para CVRP y VRPTW.

Para las instancias de CVRP utilizaremos las instancias de Homberger modificadas, estas consistirán en eliminarle las ventanas horarias, esto genera que de las 300 instancias obtengamos 30 de prueba, pues la diferencia entre las instancias de un mismo tipo (por ejemplo entre C_2_1 y C_2_2 son las ventanas horarias). Para el VRPTW utilizaremos las instancias de Homberger sin ninguna modificación.

Para determinar la cantidad de vehículos para las instancias del CVRP, se decide utilizar la cota mínima posible es decir:

$$k := \left\lceil \frac{\text{demanda total}}{\text{capacidad total}} \right\rceil$$

Donde demanda total hace referencia a la suma de las demandas de los nodos y capacidad total a la capacidad máxima de un vehículo.

Para el caso del VRPTW se utilizará la cantidad de vehículos de las best known solution (obtenidas de [46]).

5.2. Sobre los experimentos

Como ya se menciono anteriormente, la clusterización inicial para la heurística fue considerada como dada, esto significa que se requiere pensar en una. Dado que el algoritmo fue pensando para matrices de distancia arbitrarias, no necesariamente euclideanas, utilizar un algoritmo como Kmeans que se basa en calcular centroides no se vuelve factible, pues no necesariamente va a existir un valor para dicho punto, dado esto utilizaremos un algoritmo de Kmedoides, el cual tiene la bondad de ser utilizable en solo los puntos. Dentro de los algoritmos de Kmedoides uno de los algoritmos clasicos, y el primero en atacar el problema, se encuentra PAM. Este algoritmo presentado en Rousseeuw y Kaufman [30] consiste en

buscar los mejores representantes del problema mediante un proceso iterativo. Utilizaremos una versión mejorada del algoritmo llamda FasterPam que fue presentada por Schubert y Rousseeuw [47], este algoritmo consiste en realizar una modificación al algoritmo Pam lo cual le permite solventar uno de los problemas de velocidad que tiene la versión clásica.

En términos del ruteo compararemos cuatro versiones distintas, que se encuentran resumidos en la Tabla 5.1. Llamaremos *Heurística* al algoritmo propuesto en el trabajo, es decir a clusterizar, rutear cada cluster y juntar las soluciones para una ultima búsqueda local con la instancia junta. Diremos *Solo clustering* cuando hablemos de realizar el mismo proceso de *Heurística* pero utilizaremos como algoritmo de clustering solamente FasterPam, es decir solo realizar la clusterización inicial. El nombre *SimpliRoute* consistirá en rutear la instancia completa con el algoritmo de la empresa SimpliRoute. Por ultimo, *Best know solution* se referira a utilizar los resultados reportados como la mejor solución en [46].

Tabla 5.1: Tipos de ruteos utilizados

Nombre	Descripción
Heurística	Resolver el problema con la heurística planteada.
Solo clustering	Utilizar FasterPam en vez de la clusterización planteada.
SimpliRoute	Resolver la instancia completa con el solver de SimpliRoute.
Best know solution	Las mejores soluciones reportada.

5.3. Aproximación tiempo en ruta

5.3.1. CVRP

Se realizo una regresión lineal para aproximar el tiempo en ruta utilizando como variables explicativa el peso del MST de los nodos de la ruta, y la distancia promedio al depot: predicción de tiempo $= \theta_0 + \theta_1$ peso del MST $+ \theta_2$ distancia promedio al depot.

Utilizando solo el peso del MST para aproximar el resultado se obtiene un MAPE de 22.99 % entre todas las instancias, mientras que al realizar la regresión lineal se obtiene un $R^2 = 0.999$ y MAPE de 1.55 %. Las mejoras por instancia se puede apreciar en la Tabla 5.2, como se puede apreciar todos los casos presentan mejoría y en la mayoría es considerable. Antes de la regresión las instancias $R1$ y $RC1$ presentaban un valor superior a 11 % de mape y posteriormente todas ellas disminuyen a menos de 3 %, incluso en las instancias C1 y C2 donde se obtenía un buen aproximado se obtienen mejoras considerables. Cabe destacar que los resultados de la aproximación llegan a ser competitiva con los resultados presentados en Figliozzi [34].

5.3.2. VRPTW

Para el caso con ventanas horarias realizamos el algoritmo propuesto en Código 3.4 para estimar el tiempo en ruta, que de manera greedy busca insertar la mayor cantidad de puntos

Tabla 5.2: MAPE (%) usando solo el peso del MST con tiempo de servicio

Instancia	Sin regresión	Con regresión	Instancia	Sin regresión	Con regresión
C1_2	3.20	0.77	R2_6	6.95	2.31
C2_2	1.32	0.52	RC1_6	15.18	2.24
R1_2	11.83	2.19	RC2_6	7.36	2.16
R2_2	6.51	2.31	C1_8	6.41	0.65
RC1_2	11.61	2.75	C2_8	2.51	0.51
RC2_2	5.68	2.48	R1_8	16.65	1.83
C1_4	4.00	0.63	R2_8	6.85	2.38
C2_4	1.46	0.38	RC1_8	17.45	1.84
R1_4	13.10	2.05	RC2_8	6.44	3.41
R2_4	4.50	2.37	C1_10	7.80	0.67
RC1_4	13.65	1.95	C2_10	3.06	0.54
RC2_4	4.59	2.94	R1_10	18.12	1.60
C1_6	4.72	0.61	R2_10	5.86	4.36
C2_6	2.23	0.33	RC1_10	18.48	1.80
R1_6	15.45	1.59	RC2_10	9.43	3.07

mediante un orden de las ventanas horarias. Cabe destacar que es necesario hacer una modificación sobre lo que se hizo para el CVRP pues si se vuelve a entrenar el modelo utilizado anteriormente con estas instancias se obtiene un MAPE de 53.97 % con un R^2 de 0.315, esto es natural pues no se esta considerando los tiempos de espera ni la compatibilidad de las ventanas horarias dentro de este calculo.

Si utilizamos el algoritmo propuesto obtenemos un MAPE de 9.16 %, lo cual lo vuelve una buena aproximación de lo que sería el tiempo en ruta. Se intento mejorar los resultados generando una regresión lineal con la cantidad de puntos que se determinaron no ruteables, pero empeoro la predicción luego para mejorar la predicción habría que realizar una selección de propiedades mas detallada como lo que se presenta en Figliozzi [34], a pesar de esto se decide utilizar este algoritmo por que genera buenos resultados para una aproximación.

5.4. Experimentos Numéricos sobre restricción de aristas

Como ya se menciono al momento de clusterizar es posible realizar una restricción de la cantidad de aristas utilizadas en el algoritmo y vimos que para el MST esto puede incluso no disminuir la optimalidad. A nivel de clustering o cuando no se cumplen las hipótesis de los teorems demostrados es complejo de evaluar de manera teóricamente como se modifica el optimo, pues se traduce en entender como es que se modifican las conexiones tanto a nivel de MST y del clustering y como esto afecta la función objetivo de balanceo.

Ya vimos que si eliminamos las $n_{clusters}-1$ aristas mas pesadas del MST del grafo completo obtenemos una solución óptima para la zonificación y gracias al capitulo anterior sabemos que podemos saber el peor caso de la cantidad de arcos necesarios para poder recuperar la solución, bajo ciertas hipótesis dependiendo del contexto.

De esta manera nos gustaría entender que efecto tiene sobre la la función objetivo y la velocidad si realizamos la reducción de aristas. Para esto realizaremos un estudio numérico para distintos valores de K vecinos donde se evaluarán los efecto, para estos experimentos tomaremos un K que sea proporcional al número de puntos, es decir

$$K = K_n = \lceil \alpha n \rceil$$

con $\alpha \in (0, 1]$, de esta manera α puede pensarse como el porcentaje de aristas que queremos mantener conectadas. Analizaremos los caso de $\alpha \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$.

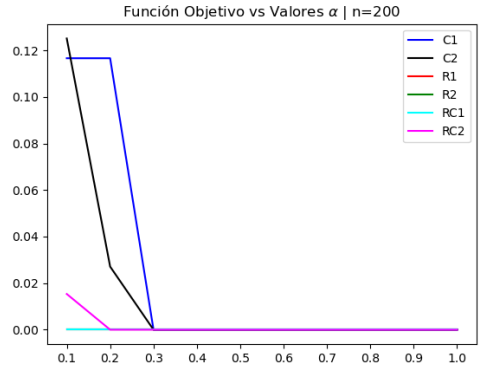
A continuación y en el Anexo D presentaremos distintos gráficos con respecto a la función objetivo y el tiempo de ejecución de la clusterización en función del valor de α seleccionado. Para entender el comportamiento de las soluciones graficaremos la diferencia relativa con respecto a utilizar $\alpha = 1$, es decir si denotamos por y_α a la función objetivo utilizaremos la forma de la 5.1 y para el tiempo de ejecución realizaremos lo mismo.

$$\frac{y_\alpha - y_1}{y_1} \tag{5.1}$$

Lo anterior significa que un numero positivo es que aumento la función objetivo mientras que uno negativo es que disminuyo, análogo para el tiempo de ejecución, por tanto el eje Y corresponderá a una proporción de aumento o disminución de la función en comparación al valor de referencia obtenido con $\alpha = 1$.

5.4.1. CVRP

En la Figura 5.1 se puede observar el efecto que tiene en la función objetivo y en los tiempos de ejecución para las instancias de tamaño 200 puntos, es posible notar como rápidamente (al considerar el 30 %) se llega a un punto donde se deja de perder optimalidad y la velocidad llega a un punto de ser al menos 50 % más rápido. Algo relevante es que si se observa el caso de tamaño 1000 (Figura 5.2) la elección de α ya no se vuelve relevante para la función objetivo pues los cambios son infimos y el efecto considerable es sobre la velocidad.



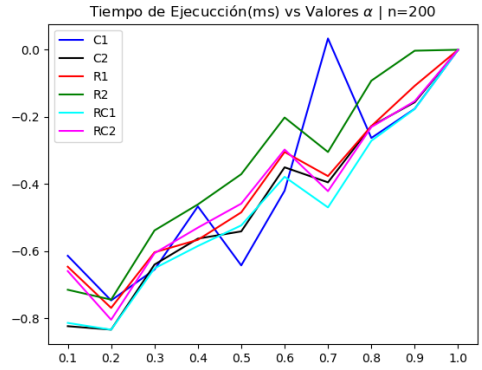
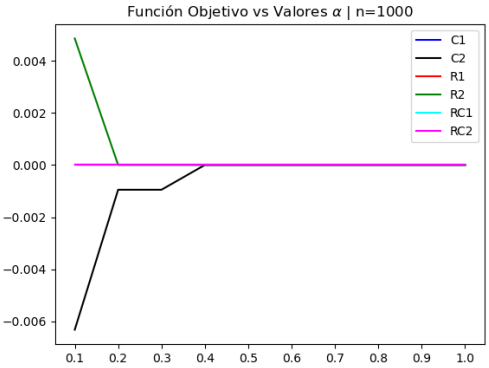


Figura 5.1: Gráficos instancias tamaño 200



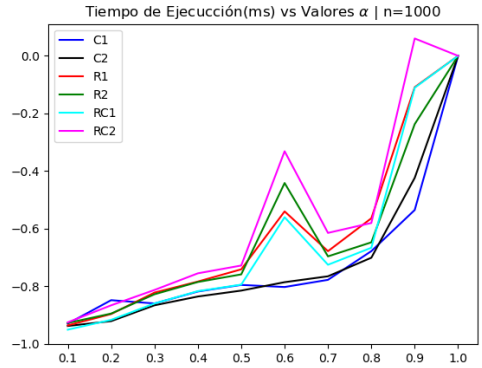


Figura 5.2: Gráficos instancias tamaño 1000

Otro punto destacable es que a veces considerar α pequeño puede ayudar a salir de óptimos locales que no son realmente beneficiosos, por ejemplo para el caso de $n = 400$ se aprecia claramente este efecto, Figura 5.3. Este comportamiento observado se aprecia para todos los tamaños de instancias, es decir valores pequeños de α, tales como $\{0.1, 0.2, 0.3\}$, entregan resultados casi óptimos o óptimos (en comparación a $\alpha = 1$) con una reducción del tiempo de ejecución considerable, para ver el resto de gráficos mirar el Anexo D.1.

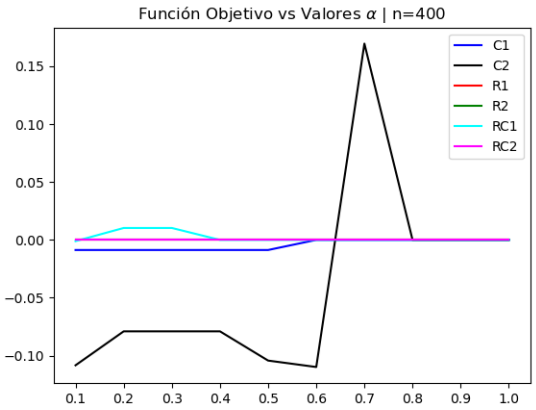


Figura 5.3: Función objetivo tamaño 400

5.4.2. VRPTW

Para este caso se vuelve más complejo la forma de representar la información pues son 10 instancias por cada tipo y tamaño. Para $n = 200$ y $n = 1000$ se muestra la primera instancia de cada tipo en Figura 5.4 y 5.5, respectivamente.

Es posible notar como se tiene un comportamiento similar a lo ya observado, existe poca o nula perdida de optimalidad, a veces incluso obteniendo mejores métricas. Lo más relevante a mencionar es que se necesita un α levemente mayor que en el caso anterior para alcanzar los mismos resultados que $\alpha = 1$, para observar este comportamiento en más detalle con el resto de casos ver Anexo D.2.

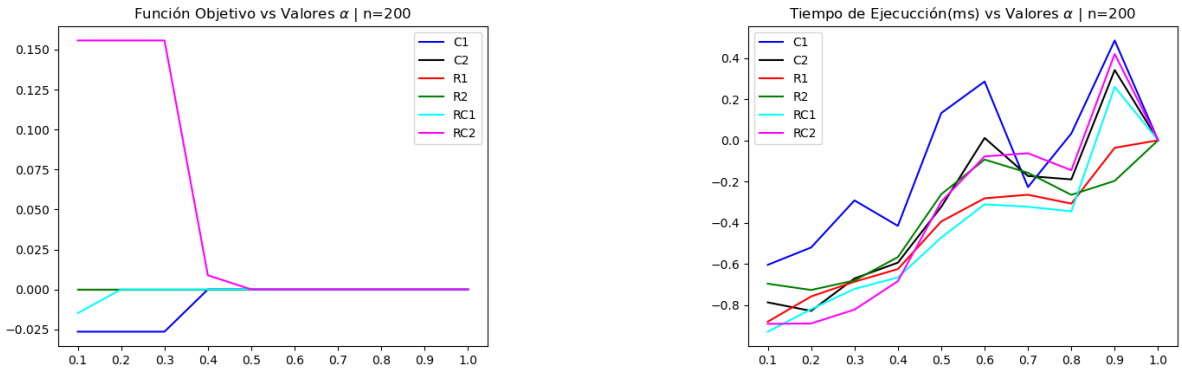


Figura 5.4: Gráficos primera instancia tamaño 200

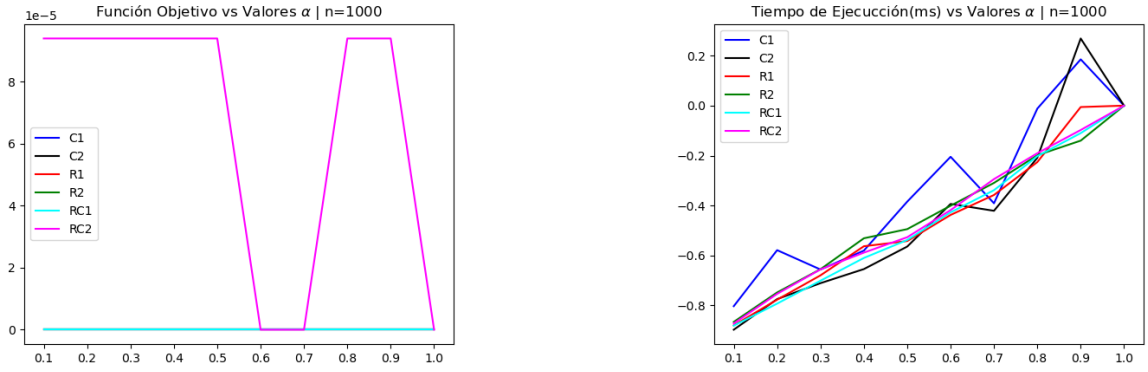


Figura 5.5: Gráficos primera instancias tamaño 1000

5.5. Experimentos Numéricos Optimización VRP

A continuación se presentan los experimentos numéricos sobre la optimización tanto para el CVRP como el VRPTW. Realizaremos tres comparaciones relevantes: *Solo clustering* vs *Heurística*, *SimpliRoute* vs *Heurística* y *Best know solution* vs *Heurística*. Es importante mencionar que este ultimo solo se realizara para el VRPTW, otra consideración relevante es que cuando se compare *SimpliRoute* vs *Heurística* y *Best know solution* vs *Heurística* se realizara una elección lexicográfica entre *Heurística* y *Solo clustering* que consistirá en quedare con la opción que inserte más puntos y en caso de que ambas estrategias inserten más se elige aquella que mejore la métrica de balanceo, con este fin se busca que el resultado sea (en el peor de los casos) tan bueno como solo hacer una clusterización.

5.5.1. CVRP

Solo clustering vs Heurística

En este caso de 30 instancias en 14 se insertan los mismos, 15 se insertan más clientes y en 1 se insertan menos. Por otro lado, en términos de la métrica de balanceo 24 de 30 instancias disminuye, de las 6 que aumentan en la mitad se llegan a insertar más clientes, además cuando se disminuye (en promedio) es de un 31 % aproximadamente, para representar esto presentamos la Figura 5.6 en este gráfico (y todos aquellos que utilicen las mismas variables en otras secciones) se utiliza lo siguiente:

$$x = \frac{\text{Nodos Insertados Heurística} - \text{Nodos Insertados Solo Clustering}}{\text{Nodos Insertados Solo Clustering}}$$
$$y = \frac{\text{Objetivo Balanceo Tiempo Heurística} - \text{Objetivo Balanceo Solo Clustering}}{\text{Objetivo Balanceo Solo Clustering}}$$

lo anterior significa que si el eje x es positivo (negativo) se insertaron más (menoss) clientes, mientras que si el y es positivo (negativo) la función objetivo aumento (disminuyo) su valor, esto implica que en el mejor de los casos nos interesa eje x positivo e y negativo, pero dado que insertar mas clientes no necesariamente nos permite mejorar la función objetivo de balanceo de tiempo en general estar en el cuadrante I y IV es un buen resultado, mientras

que si $x = 0$ ahi es relevante que el valor de y sea negativo.

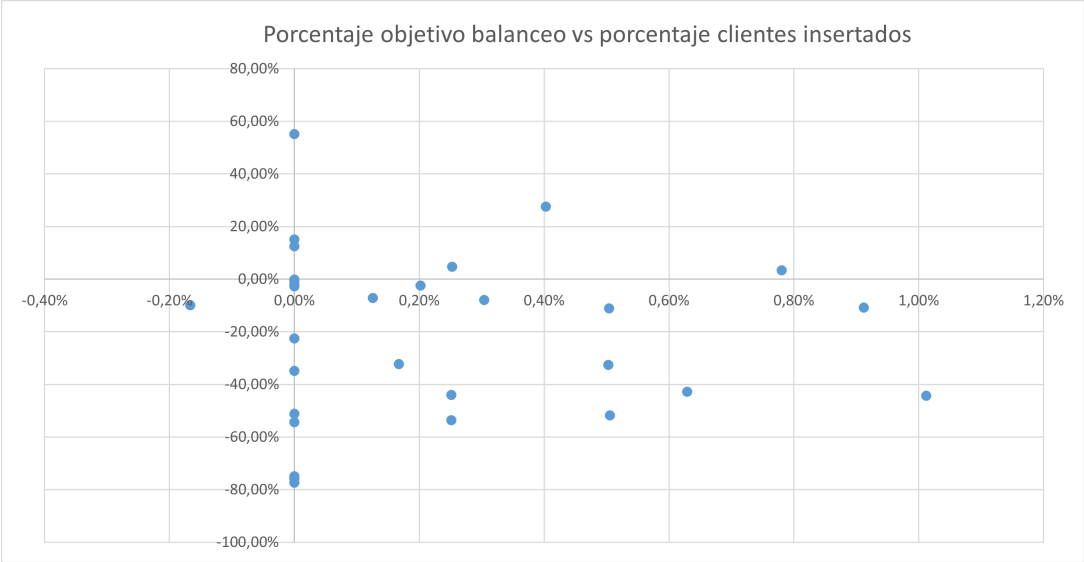


Figura 5.6: Porcentaje objetivo balanceo vs porcentaje clientes insertados

En términos de optimalidad de la solución como problema de VRP, si consideramos que la función objetivo corresponde a la suma total de la distancia recorrida, en promedio esta disminuye un 2 % (es decir es un 2 % más óptima), tal como se aprecia en la Figura 5.7.

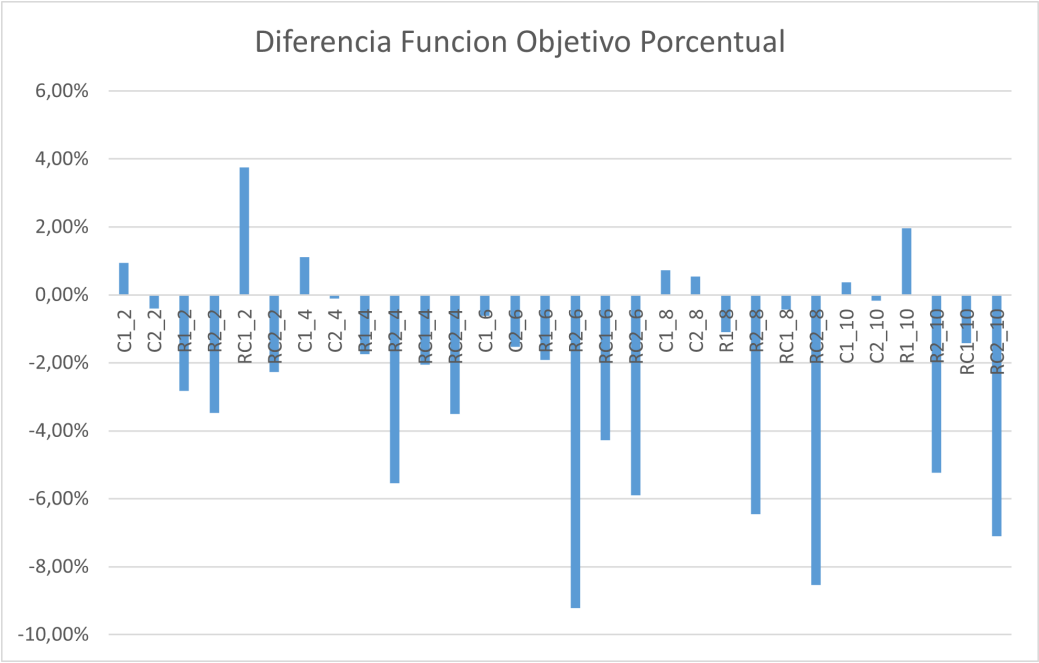


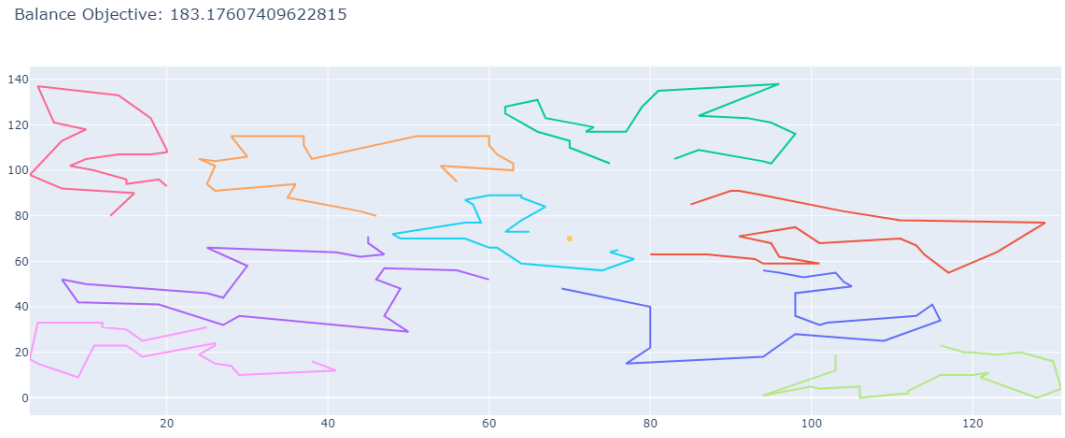
Figura 5.7: Diferencia porcentual función objetivo por instancia

Es importante mencionar que dado que dentro del algoritmo la función objetivo clásica es igual de importante que la de balanceo, por tanto observar el gráfico de la Figura 5.7 es importante para entender que tanto se esta perdiendo en optimalidad clásica por balancear,

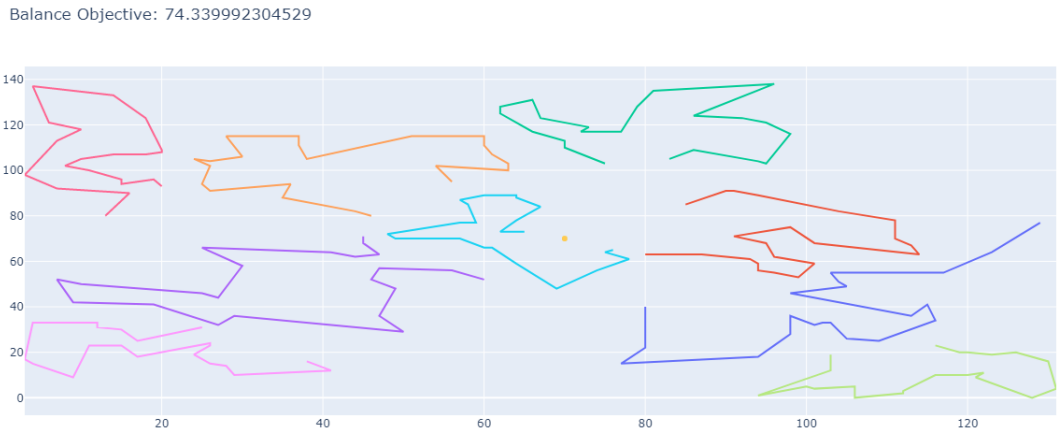
pues si este valor aumenta considerablemente en comparación al balanceo se puede volver una solución no deseada.

La discusión anterior nos muestra que el método propuesto no solo se es capaz de considerar una mejor inserción de clientes y de balanceo que solo una clusterización, sino que también el algoritmo es capaz de encontrar soluciones que son más óptimas y más balanceadas.

Para ver los efectos del algoritmo mostraremos la solución de dos instancias y el valor de la función objetivo de balanceo, esto lo realizaremos para $C2_2$ y $R1_2$, es decir las instancias $C2$ y $R1$ de 200 nodos.



(a) Solo clustering

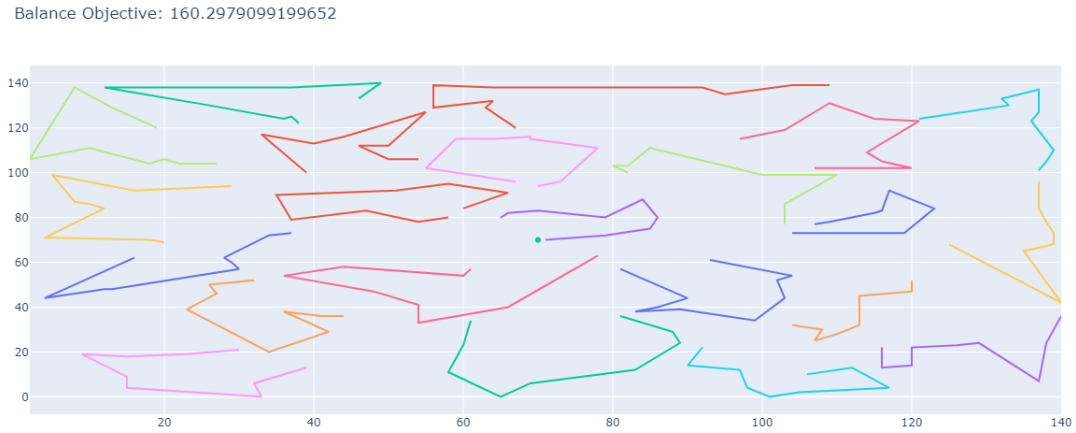


(b) Heurística.

Figura 5.8: Instancia $C2\_2$

Se puede apreciar en Figura 5.8 como es que mediantes leve cambios de puntos es posible disminuir considerablemente (por más del doble) la métrica sobre balanceo. Mientras que en 5.9 es posible observar como es que realizando movimientos de puntos entre rutas cercanas es

posible disminuir la función objetivo y generar que cluster que antes no eran factibles ahora lo sean logrando insertar más puntos, esto se puede apreciar con el punto $(120, 60)$ que no fue ruteado en *Solo clustering* pero si con la *Heurística*.



(a) Solo clustering

(b) Heurística.

Figura 5.9: Instancia $R1\_2$

SimpliRoute vs Heurística

En este caso obtenemos que en 28 de las 30 instancias se insertan más o la misma cantidad de clientes, de esos 28 caso en 20 de ellos se logra aumentar la cantidad de clientes insertados. En términos de la métrica de balanceo en 28 de las instancias disminuye. Lo anterior se puede apreciar en la Figura 5.10.

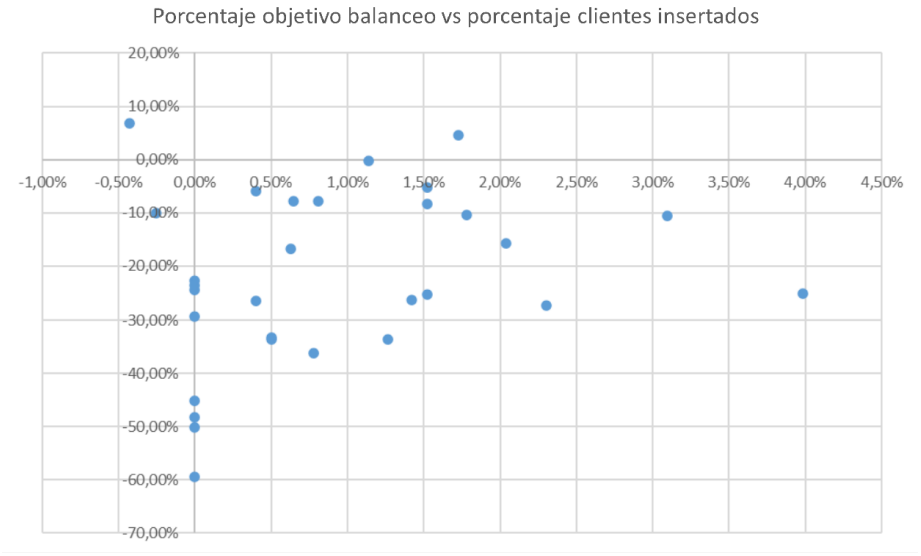


Figura 5.10: Porcentaje objetivo balanceo vs porcentaje clientes insertados

Es importante mencionar que la variación del tiempo en ruta, de ahora en adelante es lo que se entenderá como optimalidad clásica, se presenta una variación del $\pm 1\,\%$, esto implica cómo es posible buscar soluciones más balanceadas e incluso con mejor inserción sin tener que realizar un alto trade off con la optimalidad clásica, cabe destacar que estos resultados pueden variar dependiendo de que tan óptimo sea el solver utilizado.

5.5.2. VRPTW

Solo clustering vs Heurística

En este caso el $68\,\%$ de las instancias aumentan su inserción y el $78.67\,\%$ disminuye la métrica de balanceo (ver Figura 5.11), es importante destacar que en promedio la métrica disminuye un $14.06\,\%$ y, en los casos que disminuye, se transforma hasta $24.36\,\%$. En términos de optimalidad, si consideramos nuevamente la distancia recorrida, en promedio aumenta un $3\,\%$.

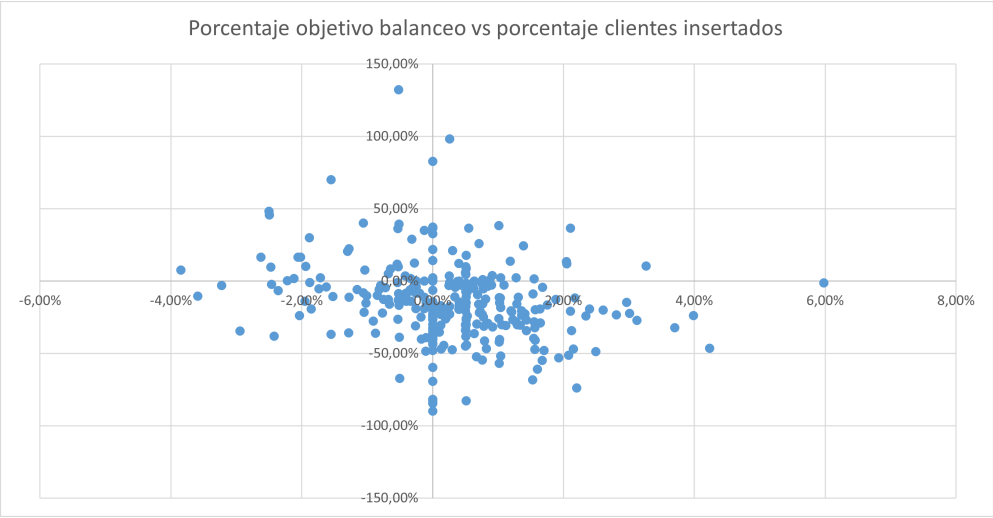
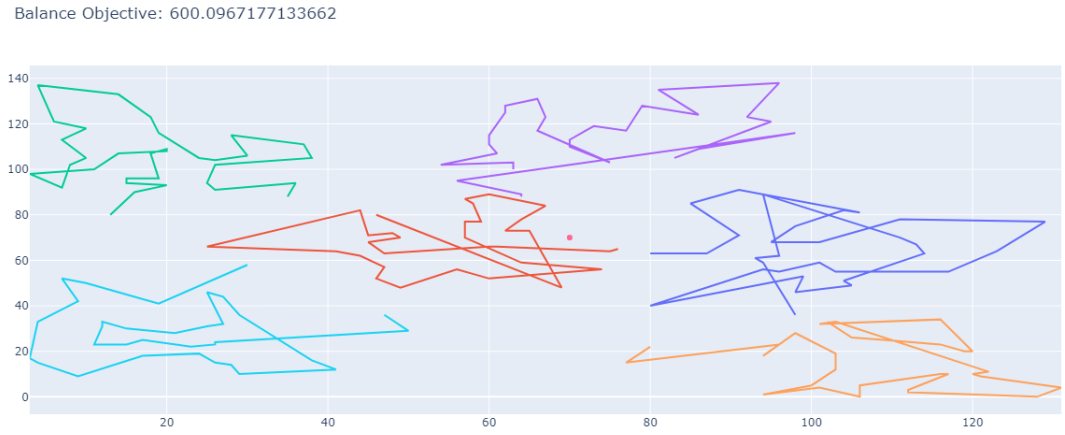


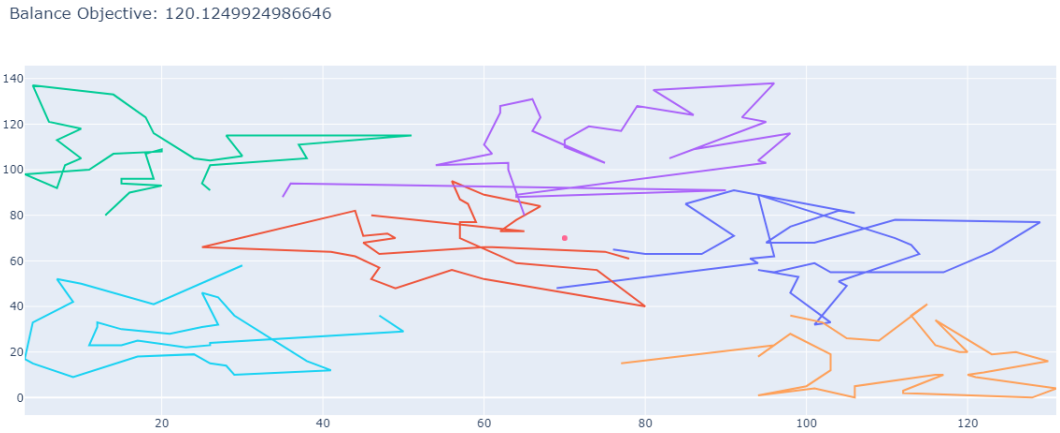
Figura 5.11: Porcentaje objetivo balanceo vs porcentaje clientes insertados

Como se puede apreciar en general el proceso de clusterización ayuda a poder aumentar los clientes insertados y disminuir la métrica de balanceo, luego aplicando un máximo entre la solución sin el proceso completo de clusterización y este proceso es posible siempre tener como base no realizar el proceso, es decir nunca empeoraremos en función a solo clusterizar y en general se tendera a mejorar el balance sin perjudicar la métrica de optimalidad.

Para mostrar comportamientos relevantes a tener en consideración se mostraran las rutas de $C2\_2\_1$ y $RC2\_2\_2$, Figura 5.12 y 5.13 respectivamente.



(a) Solo clustering

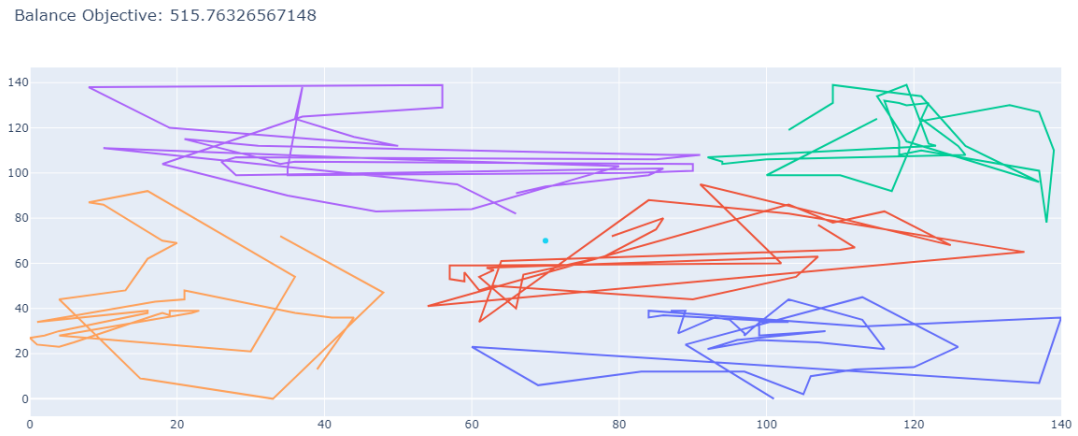


(b) Heurística.

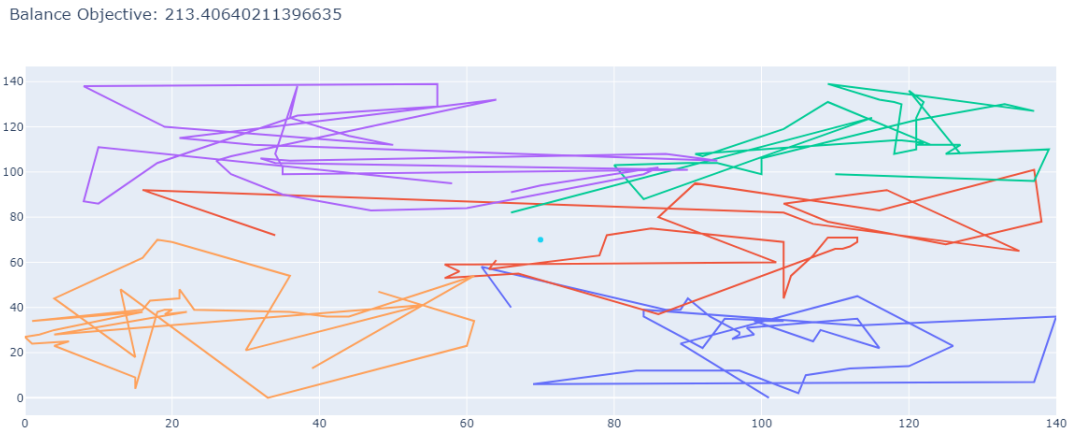
Figura 5.12: Instancia $C2\_2$

Como se puede apreciar en la Figura 5.12 sigue ocurriendo que mediante pequeños intercambios entre ruta se logra disminuir la métrica de balanceo por aproximadamente 5 veces. En la Figura 5.13 se aprecia un comportamiento más interesante, si se observa la ruta de color rojo termina cruzando todo el mapa esto se atribuye a dos puntos. El primero es por ser la métrica de balanceo el rango es posible que se intente aumentar el largo de la ruta con el fin de aumentar el valor mínimo, esto puede ocurrir tanto al momento de clusterizar como

posterior a juntar las soluciones. El segundo es debido a los intercambio permitidos con los vecinos del cluster rojo y morado son vecinos se tiene que puntos que estén en los extremos de los clusters evalúen intercambiarse, lo cual puede ser beneficioso para las métricas pero no para el resultado deseado. Dado lo anterior, si estos efectos se desean evitar puede ser beneficioso estudiar el efecto de ponderar las función objetivo, redefinir la noción de vecinos para el cluster y/o utilizar funciones que sean monótonas como el máximo o máximo lexicográfico, ver Matl et al. [4] para más detalle en el tipo de funciones de balanceo y sus características.



(a) Solo clustering



(b) Heurística.

Figura 5.13: Instancia $R1\_2$

SimpliRoute vs Heurística

Al comparar los resultados se obtiene que en 297 instancias se insertan más o la misma cantidad de clientes, siendo 2 de ellas las que insertan la misma cantidad. Mientras que en términos de la métrica asociada al balanceo en 277 se disminuye y 33 aumenta, esto se puede apreciar en el gráfico de la Figura 5.14.

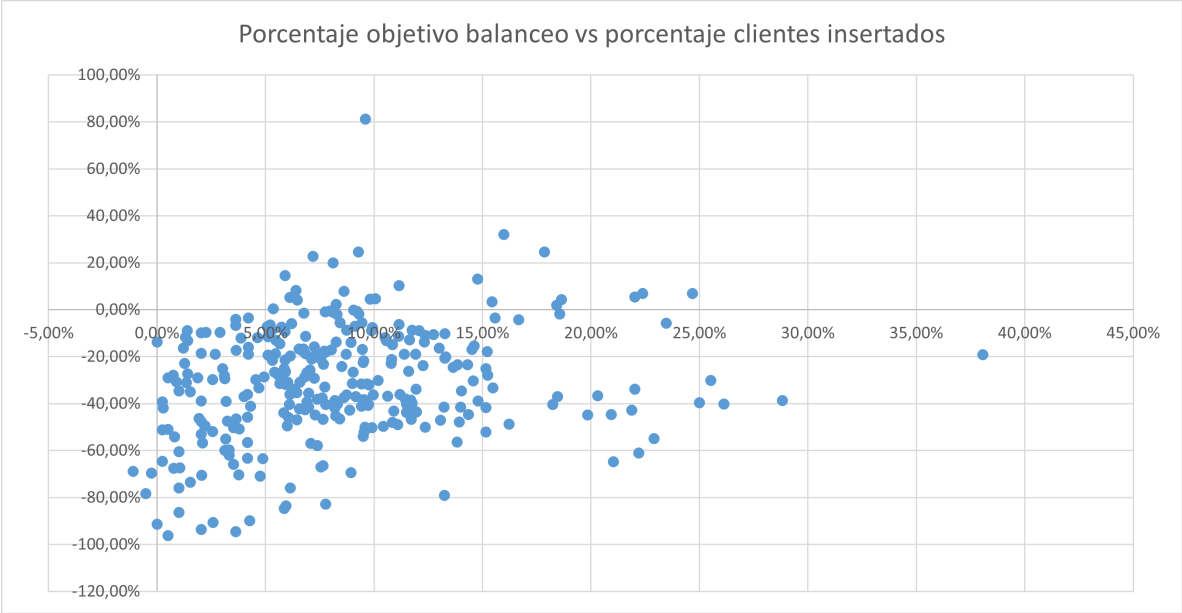


Figura 5.14: Porcentaje objetivo balanceo vs porcentaje clientes insertados

Se tiene que en promedio la metrica asociada al balanceo disminuye un 30.91 %. Si estudiamos las métricas de optimalidad clásicas la tendencia es que aumente en promedio un 9.89 %, esto puede apreciarse en el gráfico de caja de bigote de la Figura 5.15.

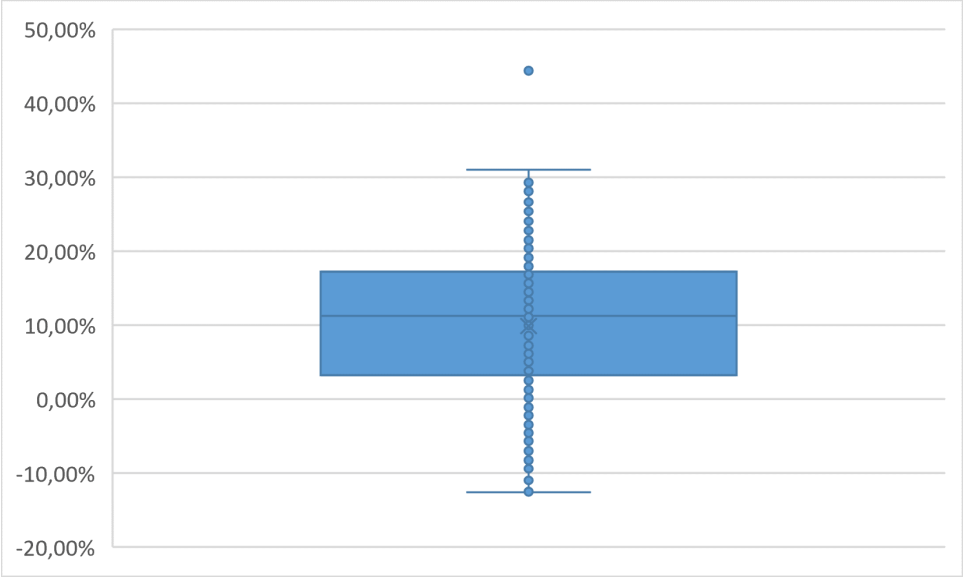


Figura 5.15: Porcentaje aumento función objetivo clásica

De lo anterior se puede apreciar cómo es que si logra generar un proceso con mejor niveles de inserción y balanceo y esto incurre en una perdida de optimalidad clásica. Cabe destacar que al insertar más clientes es posible que la perdida de optimalidad de la función objetivo clásica se deba a esto, a pesar de lo anterior se logra disminuir el balanceo en el orden de 3 veces más que el aumento ocurrido, lo cual termina siendo beneficioso para la función objetivo trabajada, que como ya se menciono consiste en la suma del balanceo y de la función clásica.

Best known solution vs Heurística

En esta parte obtenemos la Tabla 5.3 de resultados sobre la métrica de balanceo.

Tabla 5.3: Tabla Métrica Balanceo

Métrica Balanceo	Menor	Igual	Mayor
Porcentaje De Instancias	83.00 %	0.67 %	16.33 %

En general la métrica de balanceo disminuye, lo que significa que el proceso, en efecto, está haciendo lo deseado. En este caso existe un detalle y es que no en todos los casos se logran insertar el 100 % de los clientes, si se realiza ese filtro se obtiene la Tabla 5.4.

Tabla 5.4: Tabla Métrica Balanceo - Instancias Completas

Métrica Balanceo	Menor	Igual	Mayor
Porcentaje De Instancias	93.48 %	4.35 %	2.17 %

Lo anterior implica que incluso cuando se logra insertar todos los clientes de la instancia en la mayoría de los casos se logra disminuir la métrica de balanceo, en promedio el tradeoff entre optimalidad y balanceo ocurre en un ratio en que ocurre es cercano al $1:1$, es decir si es que la función objetivo clásica aumento un 5 % se obtuvo una disminución de 5 % en la métrica de balanceo. Otro punto interesante es que en la mayoría de los casos si se presenta una disminución de la métrica de balanceo utilizado.

5.6. Conclusiones de la heurística

En términos de velocidad y optimalidad se tiene que es posible utilizar una lógica de vecinos más cercanos para aumentar considerablemente la velocidad, puede llegar a ser un 80 % más rápido, sin realizar un sacrificio considerable de optimalidad en la función objetivo del clustering, es más existen casos donde se llega a mejores resultados.

Por otro lado, se puede observar cómo el proceso si logra su cometido, en general se disminuye la métrica de balanceo y logra insertar mas clientes comparándose contra un approach naive. Cuando se compara contra los resultados de la empresa en general se logran mejorar las métricas de balanceo e inserción a cambio de perjudicar un poco la metrica de suma de tiempo en ruta. Cuando se comparar contra las Best know solutions es posible observar como se encuentran soluciones mas balanceadas, y en aquellas instancias donde se logran insertar todos los clientes el tradeoff existente entre la función de optimalidad clasica y la métrica de balanceo es $1:1$, es decir si la función de optimalidad clásica disminuye un dado porcentaje en comparación a las best known solution ese valor fue aumentando en la optimalidad asociada a la métrica de balanceo.

Capítulo 6

Conclusiones

En esta tesis se logró presentar un algoritmo de clusterización enfocado al VRP, que permite obtener soluciones con consideraciones de balanceo de tiempo en ruta. Esta metodología es aplicable para cualquier forma que utilice *Cluster First - Route Second* y tiene en consideración las restricciones de capacidades y ventanas horarias del problema. Además se generan variaciones para tener en consideración cuando las ventanas horarias son restrictivas y para cuando no.

Se analizo la velocidad de la heurística y como sus resultados se ven afectados al aplicar una lógica de reducción de aristas inspirada en las búsquedas granulares. También se comparan los resultados de las optimizaciones contra los resultados entregados al realizar un ruteo sin las consideraciones y las mejores soluciones conocidas de las instancias, obteniendo que la metodología es capaz de obtener soluciones mas balanceadas y, en el caso de las mejores soluciones conocidas, generando un tradeoff 1:1 entre optimalidad clásica y balanceo.

Se logra encontrar relaciones teóricas entre el grafo de los k vecinos más cercanos y el MST. Para el caso determinista se encuentran cotas ajustadas para el valor de k donde se puede garantizar que siempre se contiene al MST. Para el contexto probabilista se logran demostrar resultados asintóticos, en el tamaño de los nodos, donde se puede garantizar que el MST esa contenido en el grafo de los vecinos mas cercanos. Además se estudio como estos resultados son aplicables para disminuir la complejidad del algoritmo propuesto y en distintas áreas que ocupen los MSTs pues nos garantizan que con alta probabilidad no perderemos la optimalidad del árbol.

Dentro de lo realizado quedan ciertas preguntas e ideas que quedan propuestas a profundizar y analizar. Por un lado, los valores para k encontrado en el contexto probabilista no son ajustados y se pueden investigar realizar cotas que sean mas justas. Por otro lado, solamente se estudio cuando se mantiene la optimalidad, se vuelve interesante intentar un análisis similar sobre cuanta optimalidad se pierde dependiendo del valor de k. Otro análisis que interesante que queda propuesto es ver si es posible extender estas nociones, o similares, a problemas mas complejos como lo son el TSP o VRP.

Bibliografía

[1] Braekers, K., Ramaekers, K., y Van Nieuwenhuyse, I., "The vehicle routing problem: State of the art classification and review", Computers & Industrial Engineering, vol. 99, pp. 300–313, 2016, doi:https://doi.org/10.1016/j.cie.2015.12.007.

[2] Dantzig, G. B. y Ramser, J. H., "The truck dispatching problem", Management science, vol. 6, no. 1, pp. 80–91, 1959.

[3] Matl, P., Hartl, R., y Vidal, T., "Workload equity in vehicle routing: The impact of alternative workload resources", Computers & Operations Research, vol. 110, pp. 116–129, 2019, doi:https://doi.org/10.1016/j.cor.2019.05.016.

[4] Matl, P., Hartl, R. F., y Vidal, T., "Workload equity in vehicle routing problems: A survey and analysis", Transportation Science, vol. 52, pp. 239–260, 2018, doi:10.1287/trsc.2017.0744.

[5] Sivaramkumar, V., Thansekhar, M. R., Saravanan, R., y Amali, S. M. J., "Demonstrating the importance of using total time balance instead of route balance on a multi-objective vehicle routing problem with time windows", The International Journal of Advanced Manufacturing Technology, vol. 98, pp. 1287–1306, 2018, doi:10.1007/s00170-018-2346-6.

[6] Fisher, M. L. y Jaikumar, R., "A generalized assignment heuristic for vehicle routing", Networks, vol. 11, no. 2, pp. 109–124, 1981, doi:https://doi.org/10.1002/net.3230110205.

[7] Gillett, B. E. y Miller, L. R., "A heuristic algorithm for the vehicle-dispatch problem", Operations Research, vol. 22, no. 2, pp. 340–349, 1974, doi:10.1287/opre.22.2.340.

[8] Gehring, H. y Homberger, J., "A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows", en University of Jyva¨skyla, pp. 57–64, 1999.

[9] Devillers, J. y Dore, J., "Heuristic potency of the minimum spanning tree (mst) method in toxicology", Ecotoxicology and Environmental Safety, vol. 17, no. 2, pp. 227–235, 1989, doi:https://doi.org/10.1016/0147-6513(89)90042-0.

[10] Tewarie, P., van Dellen, E., Hillebrand, A., y Stam, C., "The minimum spanning tree: An unbiased method for brain network analysis", NeuroImage, vol. 104, pp. 177–188, 2015, doi:https://doi.org/10.1016/j.neuroimage.2014.10.015.

[11] Naidoo, K., "Mistree: a python package for constructing and analysing minimum spanning trees", Journal of Open Source Software, vol. 4, no. 42, p. 1721, 2019, doi:10.21105/joss.01721.

[12] Laporte, G., "Fifty years of vehicle routing", Transportation science, vol. 43, no. 4, pp. 408–416, 2009.

[13] Vidal, T., Crainic, T. G., Gendreau, M., y Prins, C., "Heuristics for multi-attribute vehicle routing problems: A survey and synthesis", European Journal of Operational

Research, vol. 231, no. 1, pp. 1–21, 2013.

[14] Gendreau, M. y Tarantilis, C. D., Solving large-scale vehicle routing problems with time windows: The state-of-the-art. Cirrelt Montreal, 2010.

[15] QI, M., DING, G., ZHOU, Y., y MIAO, L., "Vehicle routing problem with time windows based on spatiotemporal distance", Journal of Transportation Systems Engineering and Information Technology, vol. 11, no. 1, pp. 85–89, 2011, doi:https://doi.org/10.1016/S1570-6672(10)60104-3.

[16] Qi, M., Lin, W.-H., Li, N., y Miao, L., "A spatiotemporal partitioning approach for large-scale vehicle routing problems with time windows", Transportation Research Part E: Logistics and Transportation Review, vol. 48, no. 1, pp. 248–257, 2012, doi:https://doi.org/10.1016/j.tre.2011.07.001. Select Papers from the 19th International Symposium on Transportation and Traffic Theory.

[17] Melián-Batista, B., De Santiago, A., AngelBello, F., y Alvarez, A., "A bi-objective vehicle routing problem with time windows: A real case in tenerife", Applied Soft Computing, vol. 17, pp. 140–152, 2014, doi:https://doi.org/10.1016/j.asoc.2013.12.012.

[18] de Armas, J., Melián-Batista, B., Moreno-Pérez, J. A., y Brito, J., "Gvns for a real-world rich vehicle routing problem with time windows", Engineering Applications of Artificial Intelligence, vol. 42, pp. 45–56, 2015, doi:https://doi.org/10.1016/j.engappai.2015.03.009.

[19] Hautamaki, V., Karkkainen, I., y Franti, P., "Outlier detection using k-nearest neighbour graph", en Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004., vol. 3, pp. 430–433 Vol.3, 2004, doi:10.1109/ICPR.2004.1334558.

[20] Lucińska, M. y Wierzchoń, S. T., "Spectral clustering based on k-nearest neighbor graph", en Computer Information Systems and Industrial Management (Cortesi, A., Chaki, N., Saeed, K., y Wierzchoń, S., eds.), (Berlin, Heidelberg), pp. 254–265, Springer Berlin Heidelberg, 2012.

[21] Dann, E., Henderson, N. C., Teichmann, S. A., Morgan, M. D., y Marioni, J. C., "Differential abundance testing on single-cell data using k-nearest neighbor graphs", Nature Biotechnology, vol. 40, pp. 245–253, 2021, doi:10.1038/s41587-021-01033-z.

[22] Bose, P., Dujmović, V., Hurtado, F., Iacono, J., Langerman, S., Meijer, H., Sacristán, V., Saumell, M., y Wood, D. R., "Proximity graphs: E, δ, δ, χ and ω", International Journal of Computational Geometry & Applications, vol. 22, no. 05, pp. 439–469, 2012.

[23] Chazelle, B., "A minimum spanning tree algorithm with inverse-ackermann type complexity", Journal of the ACM (JACM), vol. 47, no. 6, pp. 1028–1047, 2000.

[24] Nešetřil, J., Milková, E., y Nešetřilová, H., "Otakar borůvka on minimum spanning tree problem translation of both the 1926 papers, comments, history", Discrete Mathematics, vol. 233, no. 1, pp. 3–36, 2001, doi:https://doi.org/10.1016/S0012-365X(00)00224-7. Czech and Slovak 2.

[25] Frieze, A., "On the value of a random minimum spanning tree problem", Discrete Applied Mathematics, vol. 10, no. 1, pp. 47–56, 1985, doi:https://doi.org/10.1016/0166-218X(85)90058-7.

[26] Penrose, M. D., "The longest edge of the random minimal spanning tree", The Annals of Applied Probability, vol. 7, no. 2, pp. 340 – 361, 1997, doi:10.1214/aoap/1034625335.

[27] Cooper, C. y Frieze, A., "On the connectivity of random k-th nearest neighbour graphs", Combinatorics, Probability and Computing, vol. 4, pp. 343–362, 1995, doi:10.1017/s0963548300001711.

[28] Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., y Wu, A. Y., "A local search approximation algorithm for k-means clustering", Computational Geometry, vol. 28, no. 2, pp. 89–112, 2004, doi:https://doi.org/10.1016/j.comgeo.2004.03.003. Special Issue on the 18th Annual Symposium on Computational Geometry - SoCG2002.

[29] Fränti, P. y Virmajoki, O., "On the efficiency of swap-based clustering", en Adaptive and Natural Computing Algorithms (Kolehmainen, M., Toivanen, P., y Beliczynski, B., eds.), (Berlin, Heidelberg), pp. 303–312, Springer Berlin Heidelberg, 2009.

[30] Rousseeuw, P. y Kaufman, L., "Clustering by means of medoids", en Proceedings of the statistical data analysis based on the L1 norm conference, neuchatel, switzerland, vol. 31, 1987.

[31] Christofides, N., "Worst-case analysis of a new heuristic for the travelling salesman problem", rep. tec., Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group, 1976.

[32] Zhong, C., Miao, D., y Fränti, P., "Minimum spanning tree based split-and-merge: A hierarchical clustering method", Information Sciences, vol. 181, no. 16, pp. 3397–3410, 2011, doi:https://doi.org/10.1016/j.ins.2011.04.013.

[33] Nicola, D., Vetschera, R., y Dragomir, A., "Total distance approximations for routing solutions", Computers & Operations Research, vol. 102, pp. 67–74, 2019, doi:https://doi.org/10.1016/j.cor.2018.10.008.

[34] Figliozzi, M. A., "Planning approximations to the average length of vehicle routing problems with time window constraints", Transportation Research Part B: Methodological, vol. 43, no. 4, pp. 438–447, 2009, doi:https://doi.org/10.1016/j.trb.2008.08.004.

[35] Auger, N., Jugé, V., Nicaud, C., y Pivoteau, C., "On the worst-case complexity of timsort", en 26th Annual European Symposium on Algorithms (ESA 2018), vol. 112, pp. 4–1, 2018.

[36] Atkinson, M. D., Sack, J.-R., Santoro, N., y Strothotte, T., "Min-max heaps and generalized priority queues", Commun. ACM, vol. 29, p. 996–1000, 1986, doi:10.1145/6617.6621.

[37] Chin, F. y Houck, D., "Algorithms for updating minimal spanning trees", Journal of Computer and System Sciences, vol. 16, no. 3, pp. 333–344, 1978, doi:https://doi.org/10.1016/0022-0000(78)90022-3.

[38] Nardelli, E., Proietti, G., y Widmayer, P., "Nearly linear time minimum spanning tree maintenance for transient node failures", Algorithmica, vol. 40, pp. 119–132, 2004, doi:10.1007/s00453-004-1099-9.

[39] Gaibisso, C., Proietti, G., y Tan, R. B., "Optimal MST maintenance for transient deletion of every node in planar graphs", en Lecture Notes in Computer Science, pp. 404–414, Springer Berlin Heidelberg, 2003, doi:10.1007/3-540-45071-8_41.

[40] Toth, P. y Vigo, D., "The granular tabu search and its application to the vehicle-routing problem", Informs Journal on computing, vol. 15, no. 4, pp. 333–346, 2003.

[41] David, H. A. y Nagaraja, H. N., Order statistics. John Wiley & Sons, 2004.

[42] Arratia, R. y Gordon, L., "Tutorial on large deviations for the binomial distribution", Bulletin of mathematical biology, vol. 51, no. 1, pp. 125–131, 1989.

[43] Steele, J. M., "On frieze's χ (3) limit for lengths of minimal spanning trees", Discrete Applied Mathematics, vol. 18, no. 1, pp. 99–103, 1987.

[44] Tarjan, R. E., "Efficiency of a good but not linear set union algorithm", J. ACM, vol. 22, p. 215–225, 1975, doi:10.1145/321879.321884.

[45] Graham, R. L., Rothschild, B. L., y Spencer, J. H., "Ramsey theory", vol. 20. John Wiley & Sons, 1991.

[46] "Gehring & homberger benchmark.", https://www.sintef.no/projectweb/top/vrptw/.

[47] Schubert, E. y Rousseeuw, P. J., "Fast and eager k-medoids clustering: O(k) runtime improvement of the pam, clara, and clarans algorithms", Information Systems, vol. 101, p. 101804, 2021, doi:https://doi.org/10.1016/j.is.2021.101804.

Anexos

A continuación se presentan resultados que serán de utilidad para ciertas demostraciones del anexo.

En primer lugar recordar dos resultados conocidos de las normas de $\mathbb{R}^n$.

Proposición .1 *Sea $x \in \mathbb{R}^n$ denotamos por $||x||_p$ la norma p usual, es decir, $||x||_p = \left(\sum_{i=1}^{n}|x_i|^p\right)^{\frac{1}{p}}$. En tal caso*

$$\forall p \in [1,\infty], \forall x \in \mathbb{R},\ ||x||_p \leq ||x||_1, \tag{.1}$$
$$\forall x, y \in \mathbb{R}^n\ |\langle x, y\rangle| \leq ||x||_2 \cdot ||y||_2 \tag{.2}$$

Para los siguientes resultados definiremos tres modificaciones a un vector que serán de utilidad.

Definición .1 *Sea $x \in \mathbb{R}^n$ y si $x = (x(1), ..., x(n))$ se define*

$$x_a := (x(1) - a, x(2) - a, ..., x(n) - a)$$
$$x^p := ([x(1)]^p, ..., [x(n)]^p)$$
$$x_a^p := (x_a)^p = ([x(1) - a]^p, ..., [x(n) - a]^p)$$

Es importante recalcar que escribimos $x(i)$ en vez de x_i para evitar confusión.

Un resultado de utilizad sera entender como cambia la norma de un vector si lo desplazamos por una constante.

Proposición .2 *Sea $x, y \in \mathbb{R}^n$, $p, q \geq 1$ y $a, b \in \mathbb{R}$ si es que se cumple que $x(i) \geq a\ \forall i$, $y(i) \geq b\ \forall i$ entonces se cumple que*

$$|\langle x_a^p, y_b^q\rangle| \leq (||x||_1 - an)^p(||y||_1 - bn)^q$$

Demostración. Para esto notamos que por la proposición .1 se cumple que

$$|\langle x_a^p, y_b^q\rangle| \leq ||x_a^p||_2 \cdot ||y_b^q||_2 \leq ||x_a^p||_1 \cdot ||y_b^q||_1$$

Luego si denotamos $x_a^p(i)$ como el $i-$esimo elemento de x_a^p

$$
\begin{aligned}
||x_a^p||_1 &= \sum_{i=1}^{n} |x_a^p(i)| \\
&= \sum_{i=1}^{n} (|x(i) - a|)^p \\
&= ||x_a||_p^p \\
&\leq (||x_a||_1)^p \\
&\leq (||x|| - an)^p
\end{aligned}
$$

donde la ultima desigualdad se debe a que

$$
\begin{aligned}
||x_a||_1 &= \sum_{i=1}^{n} |x(i) - a| \\
&= \sum_{i=1}^{n} x(i) - a \\
&= \sum_{i=1}^{n} x(i) - an \\
&\leq \sum_{i=1}^{n} |x(i)| - an \\
&\leq ||x||_1 - an
\end{aligned}
$$

de esta manera podemos concluir que

$$
|\langle x_a^p, y_b^q \rangle| \leq ||x_a^p||_1 \cdot ||y_b^q||_1 \leq (||x||_1 - an)^p \cdot (||y||_1 - bn)^q
$$

□

Es interesante notar que sin la hipótesis $x(i) \geq a \ \forall i$, $y(i) \geq b \ \forall i$ solo se puede garantizar $|\langle x_a^p, y_b^q| \leq (||x||_1 + an)^p (||y||_1 + bn)^q$.

Anexo A. Demostraciones Capitulo 3

A.1. Demostración Proposición 3.1

Demostración. Calcularemos la complejidad por iteración y por cada local search.

Para CluseringRelocate hay que evaluar todas las combinaciones (i, j) tales que $i \neq j$, por cada una de ellas es necesario preguntarse por el movimiento de un punto, ese movimiento consiste en quitarle el punto a i y agregárselo a j. Para el modelo CVRP esto tanto para el *ZoneMST* y *BalanceMST*, recordando que el primero incluye al depot y el segundo no obtenemos la siguiente cantidad de operaciones

$$
\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j \neq i}}^{n_{clusters}} \sum_{l=1}^{n_i - 1} \Big((n_i - 1)^2 + (n_j + 1)^2 + (n_i - 2)^2 + (n_j)^2 + \log(n_{clusters}) \Big)
$$

donde n_i corresponde a la cantidad de puntos en el cluster i incluyendo al depot por tanto $n + n_{clusters} = \sum_{i=1}^{n_{clusters}} n_i$, donde n es la cantidad de nodos. Ahora procedemos a calcular cada uno de los términos relevantes. Durante todo lo que sigue denotamos $x = (n_1, n_2, ..., n_{n_{clusters}})$, entonces $x \in \mathbb{R}^{n_{clusters}}$ y $||x||_1 = n + n_{clusters}$.

Primero calculamos el termino $\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} \log(n_{clusters})$.

$$\begin{aligned}\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} \log(n_{clusters}) &= \log(n_{clusters})(n_{clusters}-1) \sum_{i=1}^{n_{clusters}} (n_i - 1) \\ &= \log(n_{clusters})(n_{clusters}-1)n\end{aligned}$$

Ahora calculamos el termino $\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_i - 2)^2$

$$\begin{aligned}\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_i - 2)^2 &= (n_{clusters}-1) \sum_{i=1}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_i - 2)^2 \\ &= (n_{clusters}-1) \sum_{i=1}^{n_{clusters}} (n_i - 1)(n_i - 2)^2 \\ &= (n_{clusters}-1)\langle x_1, x_2^2 \rangle \\ &\leq (n_{clusters}-1)(||x||_1 - n_{clusters})(||x|| - 2n_{clusters})^2 \\ &= (n_{clusters}-1)n(n - n_{clusters})^2\end{aligned}$$

Por otro lado el termino $\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_i - 1)^2$ es análogo a lo realizado pero en vez de x_2^2 es x_1^2 es decir

$$\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_i-1)^2 \leq (n_{clusters}-1)(||x||_1-n_{clusters})(||x||_1-n_{clusters})^2 = (n_{clusters}-1)n^3$$

Para calcular $\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j)^2$ realizamos un análisis similar

$$\begin{aligned}
\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j)^2 &= \sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} (n_i-1)(n_j)^2 \\
&\leq \sum_{i=1}^{n_{clusters}} \sum_{j=1}^{n_{clusters}} (n_i-1)(n_j)^2 \\
&= n \sum_{j=1}^{n_{clusters}} n_j^2 \\
&= n \sum_{j=1}^{n_{clusters}} n_j \cdot n_j \\
&= n\langle x, x\rangle \\
&\leq n||x||_1 \cdot ||x||_1 \\
&= n(n+n_{clusters})^2
\end{aligned}$$

Para el ultimo termino $\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j+1)^2$, la diferencia es que en vez de x usamos x_{-1} de esta manera

$$\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j)^2 = n\langle x_{-1}, x_{-1}\rangle \leq n(||x||_1+n_{clusters})\cdot(||x||_1+n_{clusters}) = n(n+2n_{clusters})^2$$

Juntado todo lo anterior obtenemos.

$$\begin{aligned}
&\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} \Big((n_i-1)^2 + (n_j+1)^2 + (n_i-2)^2 + (n_j)^2 + \log(n_{clusters})\Big) \\
&\leq \log(n_{clusters})(n_{clusters}-1)n + (n_{clusters}-1)n(n-n_{clusters})^2 + (n_{clusters}-1)n^3 \\
&+ n(n+n_{clusters})^2 + n(n+2n_{clusters})^2 \\
&= 2n_{clusters}n^3 - 2n_{clusters}(n_{clusters}-4) + (n_{clusters}^3 + 4n_{clusters}^2 + (n_{clusters}-1)\log(n_{clusters}))n
\end{aligned}$$

Ahora para el modelo $VRPTW$ el tiempo esta dado por el Código 3.4, el pero caso es $O(n)$ por tanto la complejidad se absorbe por la necesidad de ordenar los clientes que se hace en $O(n\log(n))$ por tanto en la parte donde remplazamos los cálculos del *balanceMST* de usar n^2 usamos $n\log(n)$ con lo cual obtenemos:

$$\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} \Big((n_i-1)^2 + (n_j+1)^2 + (n_i-2)\log(n_i-2) + n_j\log(n_j) + \log(n_{clusters})\Big)$$

con esto nos basta calcular los nuevos términos, pues los demás sabemos calcularlos.

En primer lugar calculamos $\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_i-2)\log(n_i-2)$.

$$
\begin{aligned}
&\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_i-2)\log(n_i-2) \\
&= (n_{clusters}-1) \sum_{i=1}^{n_{clusters}} (n_i-1)(n_i-2)\log(n_i-2) \\
&\leq (n_{clusters}-1)\log(n-n_{clusters}) \sum_{i=1}^{n_{clusters}} (n_i-1)(n_i-2) \\
&= (n_{clusters}-1)\log(n-n_{clusters})\langle x_1, x_2\rangle \\
&\leq (n_{clusters}-1)\log(n-n_{clusters})(||x||_1 - n_{clusters})(||x||_1 - 2n_{clusters}) \\
&\leq (n_{clusters}-1)\log(n-n_{clusters})n(n-n_{clusters})
\end{aligned}
$$

Ahora nos falta calcular $\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} n_j \log(n_j)$.

$$
\begin{aligned}
\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} n_j\log(n_j) &= \sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} (n_i-1)n_j\log(n_j) \\
&\leq \log(n+n_{clusters}) \sum_{i=1}^{n_{clusters}} \sum_{j=1}^{n_{clusters}} (n_i-1)n_j \\
&= n(n+n_{clusters})\log(n+n_{clusters})
\end{aligned}
$$

de esta manera obtenemos

$$
\begin{aligned}
&\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} \Big((n_i-1)^2 + (n_j+1)^2 + (n_i-2)\log(n_i-2) + n_j\log(n_j) + \log(n_{clusters})\Big) \\
&\leq \log(n_{clusters})(n_{clusters}-1)n + (n_{clusters}-1)n(n-n_{clusters})^2 + n(n+n_{clusters})^2 \\
&+ (n_{clusters}-1)\log(n-n_{clusters})n(n-n_{clusters}) + n(n+n_{clusters})\log(n+n_{clusters}) \\
&= n_{clusters}n^3 + (4n_{clusters} + (n_{clusters}-1)\log(n-n_{clusters}) + \log(n_{clusters}+n) - 2n_{clusters}^2)n^2 \\
&+ (n_{clusters}^3 + n_{clusters}^2\log(n-n_{clusters}) + n_{clusters}\log(n-n_{clusters}) + (n_{clusters}-1)\log(n_{clusters}) \\
&+ n_{clusters}\log(n_{clusters}+n))n
\end{aligned}
$$

Ahora para CluseringExchange hay que evaluar el triángulo superior las combinaciones (i,j), por cada una de ellas es necesario preguntarse por el intercambio de puntos entre i,j. Para el modelo CVRP esto tanto para el *ZoneMST* y *BalanceMST*, recordando que el primero incluye al depot y el segundo no y utilizando la misma notación anterior obtenemos la siguiente cantidad de operaciones

$$\sum_{i=1}^{n_{clusters}} \sum_{j=i+1}^{n_{clusters}} \sum_{l=1}^{n_i-1} \sum_{m=1}^{n_j-1} \Big((n_i-1)^2 + n_i^2 + (n_j-1)^2 + n_j^2 + \log(n_{clusters}) + (n_i-2)^2 + (n_i-1)^2 + (n_j-2)^2 + (n_j-1)^2 \Big)$$

En primer lugar trabajaremos el termino $\sum_{i=1}^{n_{clusters}} \sum_{j=i+1}^{n_{clusters}} \sum_{l=1}^{n_i-1} \sum_{m=1}^{n_j-1} \log(n_{clusters})$.

$$\begin{aligned}
\sum_{i=1}^{n_{clusters}} \sum_{j=i+1}^{n_{clusters}} \sum_{l=1}^{n_i-1} \sum_{m=1}^{n_j-1} \log(n_{clusters}) &= \log(n_{clusters}) \sum_{i=1}^{n_{clusters}} \sum_{j=i+1}^{n_{clusters}} (n_i-1)(n_j-1) \\
&\leq \log(n_{clusters}) \sum_{i=1}^{n_{clusters}} \sum_{j=1}^{n_{clusters}} (n_i-1)(n_j-1) \\
&= n^2 \log(n_{clusters})
\end{aligned}$$

Ahora acotaremos el resto de la expresión.

$$\sum_{i=1}^{n_{clusters}}\sum_{j=i+1}^{n_{clusters}}\sum_{l=1}^{n_i-1}\sum_{m=1}^{n_j-1}\Big((n_i-1)^2+n_i^2+(n_j-1)^2+n_j^2+(n_i-2)^2+(n_i-1)^2+(n_j-2)^2$$
$$+(n_j-1)^2\Big)$$
$$=\sum_{i=1}^{n_{clusters}}\sum_{j=i+1}^{n_{clusters}}\sum_{l=1}^{n_i-1}\sum_{m=1}^{n_j-1}\Big(2(n_i-1)^2+n_i^2+2(n_j-1)^2+n_j^2+(n_i-2)^2+(n_j-2)^2\Big)$$
$$=\sum_{i=1}^{n_{clusters}}\sum_{j=i+1}^{n_{clusters}}(n_i-1)(n_j-1)\Big(2(n_i-1)^2+n_i^2+2(n_j-1)^2+n_j^2+(n_i-2)^2+(n_j-2)^2\Big)$$
$$\leq\sum_{i=1}^{n_{clusters}}\sum_{j=1}^{n_{clusters}}(n_i-1)(n_j-1)\Big(2(n_i-1)^2+n_i^2+2(n_j-1)^2+n_j^2+(n_i-2)^2+(n_j-2)^2\Big)$$
$$=\sum_{i=1}^{n_{clusters}}\sum_{j=1}^{n_{clusters}}2(n_j-1)(n_i-1)^3+\sum_{i=1}^{n_{clusters}}\sum_{j=1}^{n_{clusters}}(n_i-1)(n_j-1)n_i^2$$
$$+\sum_{i=1}^{n_{clusters}}\sum_{j=1}^{n_{clusters}}2(n_i-1)(n_j-1)^3+\sum_{i=1}^{n_{clusters}}\sum_{j=1}^{n_{clusters}}(n_i-1)(n_j-1)n_j^2$$
$$+\sum_{i=1}^{n_{clusters}}\sum_{j=1}^{n_{clusters}}(n_i-1)(n_j-1)(n_i-2)^2+\sum_{i=1}^{n_{clusters}}\sum_{j=1}^{n_{clusters}}(n_i-1)(n_j-1)(n_j-2)^2$$
$$=2\Big[\sum_{i=1}^{n_{clusters}}\sum_{j=1}^{n_{clusters}}2(n_j-1)(n_i-1)^3+\sum_{i=1}^{n_{clusters}}\sum_{j=1}^{n_{clusters}}(n_i-1)(n_j-1)n_i^2$$
$$+\sum_{i=1}^{n_{clusters}}\sum_{j=1}^{n_{clusters}}(n_i-1)(n_j-1)(n_i-2)^2\Big]$$
$$=2\sum_{i=1}^{n_{clusters}}\sum_{j=1}^{n_{clusters}}(n_i-1)(n_j-1)\left[2(n_i-1)^2+n_i^2+(n_i-2)^2\right]$$
$$=2n\sum_{i=1}^{n_{clusters}}(n_i-1)\left(2(n_i-1)^2+n_i^2+(n_i-2)^2\right)$$
$$=2n\sum_{i=1}^{n_{clusters}}(n_i-1)(n_i-2)^2+2(n_i-1)^3+(n_i-1)n_i^2$$
$$\leq 2n\left[n(n-n_{clusters})^2+2n^3+n(n+n_{clusters})^2\right]$$

Donde nos falta demostrar la ultima cota realizada

$$\sum_{i=1}^{n_{clusters}}(n_i-1)(n_i-2)^2+2(n_i-1)^3+(n_i-1)n_i^2$$
$$=\langle x_1,x_2^2\rangle+2\langle x_1,x_1^2\rangle+\langle x_1,x^2\rangle$$
$$\leq(||x||_1-n_{clusters})(||x||_1-2n_{clusters})^2+2(||x||_1-n_{clusters})(||x||_1-n_{clusters})^2$$
$$+(||x||_1-n_{clusters})(||x||_1)^2$$
$$=n(n-n_{clusters})^2+2n^3+n(n+n_{clusters})^2$$

Con lo cual obtenemos

$$\begin{aligned}
&\sum_{i=1}^{n_{clusters}} \sum_{j=i+1}^{n_{clusters}} \sum_{l=1}^{n_i-1} \sum_{m=1}^{n_j-1} \Big((n_i-1)^2 + n_i^2 + (n_j-1)^2 + n_j^2 + \log(n_{clusters}) \\
&+ (n_i-2)^2 + (n_i-1)^2 + (n_j-2)^2 + (n_j-1)^2 \Big) \\
&\leq n^2 \log(n_{clusters}) + 2n \left[n(n-n_{clusters})^2 + 2n^3 + n(n+n_{clusters})^2 \right] \\
&= 8n^4 + (4n_{clusters}^2 + \log(n_{clusters}))n^2
\end{aligned}$$

Ahora miraremos el caso del VRPTW donde nuevamente hay que considera $n\log(n)$ en vez del MST de balanceo, obteniendo:

$$\sum_{i=1}^{n_{clusters}} \sum_{j=i+1}^{n_{clusters}} \sum_{l=1}^{n_i-1} \sum_{m=1}^{n_j-1} \Big((n_i-1)^2 + n_i^2 + (n_j-1)^2 + n_j^2 + \log(n_{clusters}) + \log(n_i) + \log(n_j) \Big)$$

Los únicos términos que son relevantes calcular son los $\log(n_i) + \log(n_j)$, pues el resto de terminos ya fueron calculados en la parte anterior.

$$\begin{aligned}
\sum_{i=1}^{n_{clusters}} \sum_{j=i+1}^{n_{clusters}} \sum_{l=1}^{n_i-1} \sum_{m=1}^{n_j-1} \log(n_i) + \log(n_j) &= 2 \sum_{i=1}^{n_{clusters}} \sum_{j=1}^{n_{clusters}} \sum_{l=1}^{n_i-1} \sum_{m=1}^{n_j-1} \log(n_i) \\
&= 2 \sum_{i=1}^{n_{clusters}} \sum_{j=1}^{n_{clusters}} (n_i-1)(n_j-i)\log(n_i) \\
&\leq 2\log(n+n_{clusters})n^2
\end{aligned}$$

así tenemos la cota

$$\begin{aligned}
&\sum_{i=1}^{n_{clusters}} \sum_{j=i+1}^{n_{clusters}} \sum_{l=1}^{n_i-1} \sum_{m=1}^{n_j-1} \Big((n_i-1)^2 + n_i^2 + (n_j-1)^2 + n_j^2 + \log(n_{clusters}) + \log(n_i) + \log(n_j) \Big) \\
&\leq 2n \left[2n^3 + n(n+n_{clusters})^2 \right] + 2\log(n+n_{clusters})n^2 + n^2\log(n_{clusters}) \\
&= 6n^4 + 4n_{clusters}n^3 + (2n_{clusters}^2 + 2\log(n_{clusters}+n) + \log(n_{clusters}))n^2
\end{aligned}$$

Todos los cálculos anteriores implican que para CVRP tenemos operaciones

$$\begin{aligned}
&2n_{clusters}n^3 - 2n_{clusters}(n_{clusters}-4) + (n_{clusters}^3 + 4n_{clusters}^2 + (n_{clusters}-1)\log(n_{clusters}))n \\
&+ 6n^4 + 4n_{clusters}n^3 + (2n_{clusters}^2 + 2\log(n_{clusters}+n) + \log(n_{clusters}))n^2 \\
&= 6n^4 + 6n_{clusters}n^3 + (2n_{clusters}^2 + 2\log(n_{clusters}+n) + \log(n_{clusters}))n^2 \\
&+ (n_{clusters}^3 + 4n_{clusters}^2 + (n_{clusters}-1)\log(n_{clusters}))n - 2n_{clusters}(n_{clusters}-4)
\end{aligned}$$

Ahora considerando que $\log(x) \leq x$, tenemos que (salvo constante) la expresión es posible acotarla por $n^4 + n_{clusters}n^3 + n_{clusters}^2n^2 + n_{clusters}^3n$ y considerando que $n_{clusters} \leq n$ es acotable

por n^4. Lo anterior implica que si T es el numero de iteraciones tenemos que la complejidad esta dada por $O\left(Tn^4\right)$

Por otro lado para VRPTW tenemos

$$
\begin{aligned}
&n_{clusters}n^3 + (4n_{clusters} + (n_{clusters} - 1)\log(n - n_{clusters}) + \log(n_{clusters} + n) - 2n_{clusters}^2)n^2 \\
&+ (n_{clusters}^3 + n_{clusters}^2 \log(n - n_{clusters}) + n_{clusters}\log(n - n_{clusters}) + (n_{clusters} - 1)\log(n_{clusters}) \\
&+ n_{clusters}\log(n_{clusters} + n))n + 6n^4 + 4n_{clusters}n^3 + (2n_{clusters}^2 + 2\log(n_{clusters} + n) + \log(n_{clusters}))n^2 \\
&= 6n^4 + 5n_{clusters}n^3 + (4n_{clusters} + (n_{clusters} - 1)\log(n - n_{clusters}) + 3\log(n_{clusters} + n) \\
&+ \log(n_{clusters}))n^2 + (n_{clusters}^3 + n_{clusters}^2 \log(n - n_{clusters}) + n_{clusters}\log(n - n_{clusters}) \\
&+ (n_{clusters} - 1)\log(n_{clusters}) + n_{clusters}\log(n_{clusters} + n))n
\end{aligned}
$$

luego acotando de la misma manera que antes se vuelve a concluir la cota $O(Tn^4)$ con lo que se concluye lo pedido. □

A.2. Demostración Proposición 3.2

Utilizaremos las mismas ideas anteriores para realizar las cotas. Primero miraremos el CVRP.

Antes de ir a las iteraciones por cada local search, calculamos los precálculos. Estos son sencillos corresponde a saber agregar, eliminar y eliminar y agregar todos los puntos, notamos que para este último dado que sabemos eliminar solo basta calcular el agregar, por tanto, la cantidad de iteraciones es

$$\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} n_j + \sum_{i=1}^{n_{clusters}} n_i^2 + \sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j-1) + \sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j-1)$$
$$+ \sum_{i=1}^{n_{clusters}} (n_i-1)^2 + \sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j-2)$$

En primer lugar calculamos $\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} n_j$.

$$\begin{aligned}
\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} n_j &\leq \sum_{i=1}^{n_{clusters}} \sum_{j=1}^{n_{clusters}} \sum_{l=1}^{n_i-1} n_j \\
&= \sum_{i=1}^{n_{clusters}} \sum_{j=1}^{n_{clusters}} (n_i-1)n_j \\
&= (n+n_{clusters}) \sum_{i=1}^{n_{clusters}} (n_i-1) \\
&= n(n+n_{clusters})
\end{aligned}$$

Por otro lado tenemos $\sum_{i=1}^{n_{clusters}} n_i^2$

$$\sum_{i=1}^{n_{clusters}} n_i^2 = ||x||_2^2 \leq ||x||_1^2 = (n+n_{clusters})^2$$

Luego calculamos $\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j-1)$

$$\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j - 1) \leq \sum_{i=1}^{n_{clusters}} \sum_{j=1}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j - 1)$$

$$= n \sum_{i=1}^{n_{clusters}} \sum_{l=1}^{n_i-1} 1$$

$$= n \sum_{i=1}^{n_{clusters}} (n_i - 1)$$

$$= nn = n^2$$

de manera similar a lo ya calculado se tiene que $\sum_{i=1}^{n_{clusters}} (n_i-1)^2 \leq n^2$ y $\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j-2) \leq n(n - n_{clusters})$. De esta manera tenemos que

$$\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} n_j + \sum_{i=1}^{n_{clusters}} n_i^2 + \sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j - 1) + \sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j - 1)$$
$$+ \sum_{i=1}^{n_{clusters}} (n_i - 1)^2 + \sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j - 2)$$
$$\leq n(n + n_{clusters}) + (n + n_{clusters})^2 + n^2 + n^2 + n^2 + (n - n_{clusters})n$$
$$= 6n^2 + 2n_{clusters}n + n_{clusters}^2$$
$$= 5n^2 + (n + n_{clusters})^2$$

Ahora miraremos el Relocate, si denotamos por $\hat{i}, \hat{j}$ los clusters modificados tenemos que actualizar sus valores con lo que obtenemos

$$\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} \log(n_{clusters}) + \sum_{\substack{i=1 \\ i\neq \hat{i}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{i}} - 1) + \sum_{\substack{i=1 \\ i\neq \hat{j}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{j}} + 1) + (n_{\hat{i}} - 1)^2 + (n_{\hat{j}} + 1)^2$$
$$+ \sum_{\substack{i=1 \\ i\neq \hat{i}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{i}} - 2) + \sum_{\substack{i=1 \\ i\neq \hat{j}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{j}}) + (n_{\hat{i}} - 2)^2 + (n_{\hat{j}} - 1)^2$$

En primer lugar tenemos que

$$\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} \log(n_{clusters}) = \log(n_{clusters}) \sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} (n_i - 1)$$
$$= (n_{clusters} - 1)\log(n_{clusters}) \sum_{i=1}^{n_{clusters}} (n_i - 1)$$
$$= n(n_{clusters} - 1)\log(n_{clusters})$$

Por otro lado

$$\sum_{\substack{i=1 \\ i\neq\hat{i}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{i}} - 1) = \sum_{\substack{i=1 \\ i\neq\hat{i}}}^{n_{clusters}} (n_i - 1)(n_{\hat{i}} - 1)$$
$$\leq \sum_{i=1}^{n_{clusters}} (n_i - 1)$$
$$= (n_{\hat{i}} - 1)n$$

de manera análoga obtenemos $\sum_{\substack{i=1 \\ i\neq\hat{j}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{j}}+1) \leq (n_{\hat{j}}+1)n$, $\sum_{\substack{i=1 \\ i\neq\hat{i}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{i}}-2) \leq n(n_{\hat{i}-2})$ y $\sum_{\substack{i=1 \\ i\neq\hat{i}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} n_{\hat{j}} \leq nn_{\hat{j}}$. Con lo anterior obtenemos

$$\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} \log(n_{clusters}) + \sum_{\substack{i=1 \\ i\neq\hat{i}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{i}} - 1) + \sum_{\substack{i=1 \\ i\neq\hat{j}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{j}} + 1) + (n_{\hat{i}} - 1)^2 + (n_{\hat{j}} + 1)^2$$
$$+ \sum_{\substack{i=1 \\ i\neq\hat{i}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{i}} - 2) + \sum_{\substack{i=1 \\ i\neq\hat{j}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{j}}) + (n_{\hat{i}} - 2)^2 + (n_{\hat{j}} - 1)^2$$
$$\leq (n_{clusters} - 1)n\log(n_{clusters}) + n(n_{\hat{i}} + n_{\hat{j}}) + (n_{\hat{i}} - 1)^2 + (n_{\hat{j}} + 1)^2 + n(n_{\hat{i}} + n_{\hat{j}} - 2)$$
$$+ (n_{\hat{i}} - 2)^2 + (n_{\hat{j}} - 1)^2$$
$$\leq (n_{clusters} - 1)n\log(n_{clusters}) + n(n + n_{clusters}) + 2(n + n_{clusters})^2 + 2n^2 + (n + 2n_{clusters})^2$$
$$+ (n - n_{clusters})^2$$

Por otro lado para Exchange obtenemos

$$\sum_{i=1}^{n_{clusters}} \sum_{j=i+1}^{n_{clusters}} \sum_{l=1}^{n_i-1} \sum_{m=1}^{n_j-1} \log(n_{clusters}) + \sum_{\substack{i=1 \\ i\neq\hat{i}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} n_{\hat{i}} + \sum_{\substack{i=1 \\ i\neq\hat{j}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} n_{\hat{j}} + n_{\hat{i}}^2 + n_{\hat{j}}^2$$
$$+ \sum_{\substack{i=1 \\ i\neq\hat{i}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{i}} - 1) + \sum_{\substack{i=1 \\ i\neq\hat{j}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{j}} - 1) + (n_{\hat{i}} - 1)^2 + (n_{\hat{j}} - 1)^2$$

con esto actuando igual que antes se tiene como cota

$$\sum_{i=1}^{n_{clusters}} \sum_{j=i+1}^{n_{clusters}} \sum_{l=1}^{n_i-1} \sum_{m=1}^{n_j-1} \log(n_{clusters}) + \sum_{\substack{i=1 \\ i\neq\hat{i}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} n_{\hat{i}} + \sum_{\substack{i=1 \\ i\neq\hat{j}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} n_{\hat{j}} + n_{\hat{i}}^2 + n_{\hat{j}}^2$$
$$+ \sum_{\substack{i=1 \\ i\neq\hat{i}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{i}} - 1) + \sum_{\substack{i=1 \\ i\neq\hat{j}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{j}} - 1) + (n_{\hat{i}} - 1)^2 + (n_{\hat{j}} - 1)^2$$
$$\leq n^2 \log(n_{clusters}) + n(n + n_{clusters}) + 2(n + n_{clusters})^2 + 3n^2$$

así obtenemos en total

$$5n^2 + (n + n_{clusters})^2 + T\Big[(n_{clusters} - 1)n \log(n_{clusters}) + n(n + n_{clusters}) + 2(n + n_{clusters})^2 + 2n^2$$
$$+ (n + 2n_{clusters})^2 + (n - n_{clusters})^2 + n^2 \log(n_{clusters}) + n(n + n_{clusters}) + 2(n + n_{clusters})^2 + 3n^2\Big]$$
$$= 5n^2 + (n + n_{clusters})^2 + T\Big[(n_{clusters} - 1)n \log(n_{clusters}) + 2n(n + n_{clusters}) + 4(n + n_{clusters})^2$$
$$+ 5n^2 + (n + 2n_{clusters})^2 + (n - n_{clusters})^2\Big]$$

luego notando que $n_{clusters} \leq n$ tenemos que

$$(n_{clusters} - 1)n \log(n_{clusters}) + 2n(n + n_{clusters}) + 4(n + n_{clusters})^2 + 5n^2$$
$$+ (n + 2n_{clusters})^2 + (n - n_{clusters})^2$$
$$\leq M\left(n^2 \log(n_{clusters}) + (n + n_{clusters})^2\right)$$

con M constate, con lo que se concluye que

$$O\left(n^2 + (n + n_{clusters})^2 + T\left[n^2 \log(n_{clusters}) + (n + n_{clusters})^2\right]\right)$$

Ahora vamos con el VRPTW, en este caso en vez de la segunda parte de los cálculos hay que usar $n \log(n)$ por tanto tenemos en el pre calculo

$$\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} n_j + \sum_{i=1}^{n_{clusters}} n_i^2 + \sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j - 1)$$
$$+ \sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j - 1) \log(n_j - 1) + \sum_{i=1}^{n_{clusters}} (n_i - 1) \log(n_i - 1)$$
$$+ \sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j - 2) \log(n_j - 2)$$

luego notando que

$$\sum_{i=1}^{n_{clusters}} (n_i - 1)\log(n_i - 1) \leq \log(n) \sum_{i=1}^{n_{clusters}} (n_i - 1) = n\log(n)$$

y que

$$\begin{aligned}\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j - 1)\log(n_j - 1) &\leq \log(n) \sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j - 1) \\ &\leq \log(n) \sum_{i=1}^{n} \sum_{j=1}^{n} (n_i - 1)(n_j - 1) \\ &= n^2 \log(n)\end{aligned}$$

conocemos como acotar todos los términos pues el que usa $n_j - 2$ solo requiere cambiar los cálculos por $n - n_{clusters}$ cuando corresponda, así se cumple que

$$\begin{aligned}&\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} n_j + \sum_{i=1}^{n_{clusters}} n_i^2 + \sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j - 1) \\ &+ \sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j - 1)\log(n_j - 1) + \sum_{i=1}^{n_{clusters}} (n_i - 1)\log(n_i - 1) \\ &+ \sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j - 2)\log(n_j - 2) \\ &\leq n(n + n_{clusters}) + (n + n_{clusters})^2 + n^2 + \log(n)n^2 + n\log(n) \\ &+ (n - n_{clusters})n\log(n - n_{clusters})\end{aligned}$$

Ahora miraremos el Relocate, si denotamos por $\hat{i}, \hat{j}$ los clusters modificados tenemos que actualizar sus valores con lo que obtenemos

$$\begin{aligned}&\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} \log(n_{clusters}) + \sum_{\substack{i=1 \\ i\neq \hat{i}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{i}} - 1) + \sum_{\substack{i=1 \\ i\neq \hat{j}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{j}} + 1) + (n_{\hat{i}} - 1)^2 + (n_{\hat{j}} + 1)^2 \\ &+ \sum_{\substack{i=1 \\ i\neq \hat{i}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{i}} - 2)\log(n_{\hat{i}} - 2) + \sum_{\substack{i=1 \\ i\neq \hat{j}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{j}})\log\Big(n_{\hat{j}}\Big) \\ &+ \sum_{i=1}^{n_{\hat{i}}-2} (n_{\hat{i}} - 2)\log(n_{\hat{i}} - 2) + \sum_{i=1}^{n_{\hat{j}}} (n_{\hat{j}})\log\Big(n_{\hat{j}}\Big)\end{aligned}$$

notamos que son todas estructuras que ya hemos calculado o hay que hacer leves modificaciones así obtenemos la siguiente cota

$$\sum_{i=1}^{n_{clusters}}\sum_{\substack{j=1\\j\neq i}}^{n_{clusters}}\sum_{l=1}^{n_i-1}\log(n_{clusters})+\sum_{\substack{i=1\\i\neq\hat{i}}}^{n_{clusters}}\sum_{l=1}^{n_i-1}(n_{\hat{i}}-1)+\sum_{\substack{i=1\\i\neq\hat{j}}}^{n_{clusters}}\sum_{l=1}^{n_i-1}(n_{\hat{j}}+1)$$
$$+(n_{\hat{i}}-1)^2+(n_{\hat{j}}+1)^2+\sum_{\substack{i=1\\i\neq\hat{i}}}^{n_{clusters}}\sum_{l=1}^{n_i-1}(n_{\hat{i}}-2)\log(n_{\hat{i}}-2)+\sum_{\substack{i=1\\i\neq\hat{j}}}^{n_{clusters}}\sum_{l=1}^{n_i-1}(n_{\hat{j}})\log\Big(n_{\hat{j}}\Big)$$
$$+\sum_{i=1}^{n_{\hat{i}}-2}(n_{\hat{i}}-2)\log(n_{\hat{i}}-2)+\sum_{i=1}^{n_{\hat{j}}}(n_{\hat{j}})\log\Big(n_{\hat{j}}\Big)$$
$$\leq(n_{clusters}-1)n\log(n_{clusters})+n(n_{\hat{i}}+n_{\hat{j}})+(n_{\hat{i}}-1)^2+(n_{\hat{j}}+1)^2$$
$$+n((n_{\hat{i}}-2)\log(n_{\hat{i}}-2)+(n_{\hat{j}})\log\Big(n_{\hat{j}}\Big))+((n_{\hat{i}}-2)^2\log(n_{\hat{i}}-2)+(n_{\hat{j}})^2\log\Big(n_{\hat{j}}\Big))$$
$$\leq(n_{clusters}-1)n\log(n_{clusters})+n(n+n_{clusters})+(n-n_{clusters})^2+(n+2n_{clusters})^2$$
$$+n(n-n_{clusters})\log(n-n_{clusters})+n(n+n_{clusters})\log(n+n_{clusters})$$
$$+(n-n_{clusters})^2\log(n-n_{clusters})+(n+n_{clusters})^2\log(n+n_{clusters})$$

para Exchange se tiene

$$\sum_{i=1}^{n_{clusters}}\sum_{j=i+1}^{n_{clusters}}\sum_{l=1}^{n_i-1}\sum_{k=1}^{n_j-1}\log(n_{clusters})+\sum_{\substack{i=1\\i\neq\hat{i}}}^{n_{clusters}}\sum_{l=1}^{n_i-1}(n_{\hat{i}})+\sum_{\substack{i=1\\i\neq\hat{j}}}^{n_{clusters}}\sum_{l=1}^{n_i-1}(n_{\hat{j}})+(n_{\hat{i}})^2+(n_{\hat{j}})^2$$
$$+\sum_{\substack{i=1\\i\neq\hat{i}}}^{n_{clusters}}\sum_{l=1}^{n_i-1}(n_{\hat{i}}-1)\log(n_{\hat{i}}-1)+\sum_{\substack{i=1\\i\neq\hat{j}}}^{n_{clusters}}\sum_{l=1}^{n_i-1}(n_{\hat{j}}-1)\log\Big(n_{\hat{j}}-1\Big)$$
$$+\sum_{i=1}^{n_{\hat{i}}-1}(n_{\hat{i}}-1)\log(n_{\hat{i}}-1)+\sum_{i=1}^{n_{\hat{j}}-1}(n_{\hat{j}}-1)\log\Big(n_{\hat{j}}-1\Big)$$

de manera similar a lo anterior obtenemos

$$\sum_{i=1}^{n_{clusters}}\sum_{j=i+1}^{n_{clusters}}\sum_{l=1}^{n_i-1}\sum_{k=1}^{n_j-1}\log(n_{clusters})+\sum_{\substack{i=1\\i\neq\hat{i}}}^{n_{clusters}}\sum_{l=1}^{n_i-1}(n_{\hat{i}})+\sum_{\substack{i=1\\i\neq\hat{j}}}^{n_{clusters}}\sum_{l=1}^{n_i-1}(n_{\hat{j}})+(n_{\hat{i}})^2+(n_{\hat{j}})^2$$
$$+\sum_{\substack{i=1\\i\neq\hat{i}}}^{n_{clusters}}\sum_{l=1}^{n_i-1}(n_{\hat{i}}-1)\log(n_{\hat{i}}-1)+\sum_{\substack{i=1\\i\neq\hat{j}}}^{n_{clusters}}\sum_{l=1}^{n_i-1}(n_{\hat{j}}-1)\log\Big(n_{\hat{j}}-1\Big)$$
$$+\sum_{i=1}^{n_{\hat{i}}-1}(n_{\hat{i}}-1)\log(n_{\hat{i}}-1)+\sum_{i=1}^{n_{\hat{j}}}(n_{\hat{j}}-1)\log\Big(n_{\hat{j}-1}\Big)$$
$$\leq n^2\log(n_{clusters})+n(n_{\hat{i}}+n_{\hat{j}})+(n_{\hat{i}})^2+(n_{\hat{j}})^2+n((n_{\hat{i}}-1)\log(n_{\hat{i}}-1)+(n_{\hat{j}}-1)\log\Big(n_{\hat{j}}-1\Big))$$
$$+(n_{\hat{i}}-1)^2\log(n_{\hat{i}}-1)+(n_{\hat{j}}-1)^2\log\Big(n_{\hat{j}}-1\Big)$$
$$\leq n^2\log(n_{clusters})+n(n+n_{clusters})+(n+n_{clusters})^2+n(n\log(n))+n^2\log(n)$$
$$=n^2\log(n_{clusters})+n(n+n_{clusters})+(n+n_{clusters})^2+2n^2\log(n)$$

así obtenemos

$$
\begin{aligned}
&n(n+n_{clusters})+(n+n_{clusters})^2+n^2+\log(n)n^2+n\log(n)\\
&+(n-n_{clusters})n\log(n-n_{clusters})+T\Big(n^2\log(n_{clusters})+n(n+n_{clusters})+(n+n_{clusters})^2\\
&+2n^2\log(n)+(n_{clusters}-1)n\log(n_{clusters})+n(n+n_{clusters})+(n-n_{clusters})^2+(n+2n_{clusters})^2\\
&+n(n-n_{clusters})\log(n-n_{clusters})+n(n+n_{clusters})\log(n+n_{clusters})\\
&+(n-n_{clusters})^2\log(n-n_{clusters})+(n+n_{clusters})^2\log(n+n_{clusters})\Big)
\end{aligned}
$$

con lo que obtenemos luego utilizando que $n_{clusters} \leq n$ tenemos que

$$O\left(n^2\log(n)+(n+n_{clusters})^2+T\left[n^2\log(n)+(n+n_{clusters})^2\right]\right)$$

concluyendo lo deseado.

Anexo B. Feasibles

Al momento de hacer los feasibles siempre existen dos casos que considerar: (1) estamos en una situación infactible (2) estamos en una situación factible.

Dado lo anterior lo deseable es que si estamos en (1) no empeoremos la infactiblidad de la situación y si estamos en (2) que el cambio siga manteniendo la factiblidad, luego los feasibles se encargaran de ese proceso.

Para Capacidad diremos que un cluster es factible si es que la capacidad, cantidad de carga que acepta el vehículo asignado al cluster, es menor o igual a a la carga del cluster, suma de la carga de cada punto que pertenece al cluster.

Para tiempos de viajes del vehículo o travelTime, recordamos que para un CVRP se basa en el MST y para un VRPTW en el algoritmo generado, diremos que es factible en travelTime si es que el travelTime es menor o igual al tiempo disponible del vehículo.

Para TimeWindow llamaremos como Unfeasible a la cantidad de puntos infactibles por TW que nos entrega el algoritmo que calcula el tiempo en ruta, notar que es parte de los resultados en conjunto a travelTime.

Notar que tanto para travelTime como Unfeasible los valores utilizados en este feasible ya están pre calculadas luego es $O(1)$ la consulta.

B.1. Relocate

Código B.1: Capacity Feasible Clustering Relocate

```
Function isCapacityFeasibleClusteringRelocate(x_i, Cluster_i, Cluster_j)
    feasiblePartition_i = Cluster_i capacity feasible
    feasiblePartition_j = Cluster_j capacity feasible

    If (feasiblePartition_i AND feasiblePartition_j)
        % Solo aceptamos cambio si no sobrepasamos la capacidad del cluster j
        return Cluster_j load + x_i load <= Cluster_j capacity
    Else If (feasiblePartition_i)
        % Le estamos agregando capacidad a un cluster sobre utilizado
        return False
    Else If (feasiblePartition_j)
        If (Cluster_j capacity < Cluster_j load + x_i load)
            % Hacemos el cluster j infactible por factibilizar el cluster i
            return Cluster_j load + x_i load - Cluster_j capacity <= Cluster_i load - Cluster_i
     ↪ capacity
        End
    Else
        % Hacemos el cluster j más infactible por factibilizar el cluster i
        return Cluster_j load + x_i load - Cluster_j capacity <= Cluster_i load - Cluster_i capacity
    End
    return True
End Function
```

Código B.2: TravelTime Feasible Clustering Relocate

```
Function isTravelTimeFeasibleClusteringRelocate(x_i, Cluster_i, Cluster_j)
    isTravelTimeFeasible_i = Cluster_i travelTime feasible
    isTravelTimeFeasible_j = Cluster_j travelTime feasible
    newTravelTime_i = Cluster_i travel time sin x_i
    newTravelTime_j = Cluster_j travel time con x_i
    If (isTravelTimeFeasible_i AND isTravelTimeFeasible_j)
        return newTravelTime_i <= Cluster_i tiempo disponible AND newTravelTime_j <=
     ↪ Cluster_j tiempo disponible
    Else If (isTravelTimeFeasible_i)
        If (newTravelTime_j > Cluster_j travelTime)
            return False
        End

        If (newTravelTime_i > Cluster_i tiempo disponible)
            return newTravelTime_i - Cluster_i tiempo disponible <= Cluster_j travelTime -
     ↪ Cluster_j tiempo disponible
        End
    Else If (isTravelTimeFeasible_j)
        If (newTravelTime_i > Cluster_i travelTime)
            return False
        End

        If (newTravelTime_j > Cluster_j tiempo disponible)
            return newTravelTime_j - Cluster_j tiempo disponible <= Cluster_i travelTime -
     ↪ Cluster_i tiempo disponible
        End
    Else
        return newTravelTime_i <= Cluster_i travel time AND newTravelTime_j <= Cluster_j
     ↪ travel time
    End
    return True
End Function
```

Código B.3: TimeWindow Feasible Clustering Relocate

```
Function isTimeWindowFeasibleClusteringRelocate(x_i, Cluster_i, Cluster_j)
    numberOfUnFeasible_i = Cluster_i unFeasible
    numberOfUnFeasible_j = Cluster_j unFeasible
    newNumberOfUnFeasible_i = Cluster_i unFeasible sin x_i
    newNumberOfUnFeasible_j = Cluster_j unFeasible con x_i
    If (numberOfUnFeasible_i == 0 AND numberOfUnFeasible_j == 0)
        return newNumberOfUnFeasible_i == 0 AND newNumberOfUnFeasible_j == 0
    Else If (numberOfUnFeasible_i == 0)
        If (newNumberOfUnFeasible_j > numberOfUnFeasible_j)
            return False
        End

```

```
13      If (newNumberOfUnFeasible_i > 0)
14          return newNumberOfUnFeasible_i <= numberOfUnFeasible_j
15      End
16  Else If (numberOfUnFeasible_j == 0)
17      If (newNumberOfUnFeasible_i > numberOfUnFeasible_i)
18          return False
19      End
20
21      If (newNumberOfUnFeasible_j > 0)
22          return newNumberOfUnFeasible_j <= numberOfUnFeasible_i
23      End
24  Else
25      return newNumberOfUnFeasible_i <= numberOfUnFeasible_i AND
    ↪ newNumberOfUnFeasible_j <= numberOfUnFeasible_j
26  End
27  return True
28 End Function
```

B.2. Exchange

Código B.4: Capacity Feasible Clustering Exchange

```
1  Function isCapacityFeasibleClusteringRelocate(x_i, x_j, Cluster_i, Cluster_j)
2      feasiblePartition_i = Cluster_i capacity feasible
3      feasiblePartition_j = Cluster_j capacity feasible
4
5      newLoad_i = Cluster_i load + x_j load - x_i load
6      newLoad_j = Cluster_j load + x_i load - x_j load
7
8      If (feasiblePartition_i AND feasiblePartition_j)
9          return newLoad_i <= Cluster_i capacity AND newLoad_j <= Cluster_j capacity
10     Else If (feasiblePartition_i)
11         If (newLoad_j > Cluster_j load)
12             return False
13         End
14
15         If (newLoad_i > Cluster_i capacity)
16             return newLoad_i - Cluster_i capacity <= Cluster_j load - Cluster_j capacity
17         End
18     Else If (feasiblePartition_j)
19         If (newLoad_i > Cluster_i load)
20             return False
21         End
22
23         If (newLoad_j > Cluster_j capacity)
24             return newLoad_j - Cluster_j capacity <= Cluster_i load - Cluster_i capacity
25         End
26     Else
27         return newLoad_i <= Cluster_i load AND newLoad_j <= Cluster_j load
28     End
29 End Function
```

Código B.5: TravelTime Feasible Clustering Exchange

```
Function isTravelTimeFeasibleClusteringExchange(x_i, x_j, Cluster_i, Cluster_j)
    isTravelTimeFeasible_i = Cluster_i travelTime feasible
    isTravelTimeFeasible_j = Cluster_j travelTime feasible
    newTravelTime_i = Cluster_i travel time sin x_i y con x_j
    newTravelTime_j = Cluster_j travel time sin x_j y con x_i
    If (isTravelTimeFeasible_i AND isTravelTimeFeasible_j)
        return newTravelTime_i <= Cluster_i tiempo disponible AND newTravelTime_j <=
     ↪ Cluster_j tiempo disponible
    Else If (isTravelTimeFeasible_i)
        If (newTravelTime_j > Cluster_j travelTime)
            return False
        End

        If (newTravelTime_i > Cluster_i tiempo disponible)
            return newTravelTime_i - Cluster_i tiempo disponible <= Cluster_j travelTime -
     ↪ Cluster_j tiempo disponible
        End
    Else If (isTravelTimeFeasible_j)
        If (newTravelTime_i > Cluster_i travelTime)
            return False
        End

        If (newTravelTime_j > Cluster_j tiempo disponible)
            return newTravelTime_j - Cluster_j tiempo disponible <= Cluster_i travelTime -
     ↪ Cluster_i tiempo disponible
        End
    Else
        return newTravelTime_i <= Cluster_i travel time AND newTravelTime_j <= Cluster_j
     ↪ travel time
    End
    return True
End Function
```

Código B.6: TimeWindow Feasible Clustering Exchange

```
Function isTimeWindowFeasibleClusteringExchange(x_i, x_j, Cluster_i, Cluster_j)
    numberOfUnFeasible_i = Cluster_i unFeasible
    numberOfUnFeasible_j = Cluster_j unFeasible
    newNumberOfUnFeasible_i = Cluster_i unFeasible sin x_i y con x_j
    newNumberOfUnFeasible_j = Cluster_j unFeasible sin x_j y con x_i
    If (numberOfUnFeasible_i == 0 AND numberOfUnFeasible_j == 0)
        return newNumberOfUnFeasible_i == 0 AND newNumberOfUnFeasible_j == 0
    Else If (numberOfUnFeasible_i == 0)
        If (newNumberOfUnFeasible_j > numberOfUnFeasible_j)
            return False
        End

```

```
        If (newNumberOfUnFeasible_i > 0)
            return newNumberOfUnFeasible_i <= numberOfUnFeasible_j
        End
    Else If (numberOfUnFeasible_j == 0)
        If (newNumberOfUnFeasible_i > numberOfUnFeasible_i)
            return False
        End

        If (newNumberOfUnFeasible_j > 0)
            return newNumberOfUnFeasible_j <= numberOfUnFeasible_i
        End
    Else
        return newNumberOfUnFeasible_i <= numberOfUnFeasible_i AND
     ↪ newNumberOfUnFeasible_j <= numberOfUnFeasible_j
    End
    return True
End Function
```

B.3. Capacity Strict Feasibles

Para este caso solo se modifica el local search asociado a capacidad y se utilizan las siguientes versiones.

Código B.7: Capacity Strict Feasible Clustering Relocate

```
Function isCapacityStrictFeasibleClusteringRelocate(x_i, Cluster_i, Cluster_j)
    feasiblePartition_i = Cluster_i capacity feasible
    feasiblePartition_j = Cluster_j capacity feasible

    If (feasiblePartition_i AND feasiblePartition_j)
        % No nos interesan clusters factibles
        return False
    Else If (feasiblePartition_i)
        % Le estamos agregando capacidad a un cluster sobre utilizado
        return False
    Else If (feasiblePartition_j)
        If (Cluster_j capacity < Cluster_j load + x_i load)
            % Si el cambio no mejora estrictamente no lo realizamos
            return Cluster_j load + x_i load - Cluster_j capacity < Cluster_i load - Cluster_i
     ↪ capacity
        End
    Else
        % Si el cambio no mejora estrictamente no lo realizamos
        return Cluster_j load + x_i load - Cluster_j capacity < Cluster_i load - Cluster_i capacity
    End
    return True
End Function
```

Código B.8: Capacity Strict Feasible Clustering Exchange

```
Function isCapacityStrictFeasibleClusteringRelocate(x_i, x_j, Cluster_i, Cluster_j)
```

```
    feasiblePartition_i = Cluster_i capacity feasible
    feasiblePartition_j = Cluster_j capacity feasible

    newLoad_i = Cluster_i load + x_j load - x_i load
    newLoad_j = Cluster_j load + x_i load - x_j load

    If (feasiblePartition_i AND feasiblePartition_j)
        return False
    Else If (feasiblePartition_i)
        % Si es que no cambiamos el load infactible no nos interesa
        If (newLoad_j >= Cluster_j load)
            return False
        End

        If (newLoad_i > Cluster_i capacity)
            return newLoad_i - Cluster_i capacity < Cluster_j load - Cluster_j capacity
        End
    Else If (feasiblePartition_j)
        If (newLoad_i >= Cluster_i load)
            return False
        End

        If (newLoad_j > Cluster_j capacity)
            return newLoad_j - Cluster_j capacity < Cluster_i load - Cluster_i capacity
        End
    Else
        return newLoad_i < Cluster_i load AND newLoad_j < Cluster_j load
    End
End Function
```

B.4. TimeWindow Strict Feasibles

En este caso modificaremos aquellos feasibles asociados a TimeWindowFeasible.

Código B.9: TimeWindow Strict Feasible Clustering Relocate

```
Function isTimeWindowStrictFeasibleClusteringRelocate(x_i, Cluster_i, Cluster_j)
    numberOfUnFeasible_i = Cluster_i unFeasible
    numberOfUnFeasible_j = Cluster_j unFeasible
    newNumberOfUnFeasible_i = Cluster_i unFeasible sin x_i
    newNumberOfUnFeasible_j = Cluster_j unFeasible con x_i
    If (numberOfUnFeasible_i == 0 AND numberOfUnFeasible_j == 0)
        % No nos interesa dos clusters factibles
        return False
    Else If (numberOfUnFeasible_i == 0)
        If (newNumberOfUnFeasible_j >= numberOfUnFeasible_j)
            % Si no mejora estrictamente no realizamos el cambio
            return False
        End

        If (newNumberOfUnFeasible_i > 0)
            return newNumberOfUnFeasible_i < numberOfUnFeasible_j
```

```
        End
    Else If (numberOfUnFeasible_j == 0)
        If (newNumberOfUnFeasible_i >= numberOfUnFeasible_i)
            return False
        End

        If (newNumberOfUnFeasible_j > 0)
            return newNumberOfUnFeasible_j < numberOfUnFeasible_i
        End
    Else
        return newNumberOfUnFeasible_i < numberOfUnFeasible_i AND
     ↪ newNumberOfUnFeasible_j < numberOfUnFeasible_j
    End
    return True
End Function
```

Código B.10: TimeWindow Strict Feasible Clustering Exchange

```
Function isTimeWindowStrictFeasibleClusteringExchange(x_i, x_j, Cluster_i, Cluster_j)
    numberOfUnFeasible_i = Cluster_i unFeasible
    numberOfUnFeasible_j = Cluster_j unFeasible
    newNumberOfUnFeasible_i = Cluster_i unFeasible sin x_i y con x_j
    newNumberOfUnFeasible_j = Cluster_j unFeasible sin x_j y con x_i
    If (numberOfUnFeasible_i == 0 AND numberOfUnFeasible_j == 0)
        return False
    Else If (numberOfUnFeasible_i == 0)
        If (newNumberOfUnFeasible_j >= numberOfUnFeasible_j)
            return False
        End

        If (newNumberOfUnFeasible_i > 0)
            return newNumberOfUnFeasible_i < numberOfUnFeasible_j
        End
    Else If (numberOfUnFeasible_j == 0)
        If (newNumberOfUnFeasible_i >= numberOfUnFeasible_i)
            return False
        End

        If (newNumberOfUnFeasible_j > 0)
            return newNumberOfUnFeasible_j < numberOfUnFeasible_i
        End
    Else
        return newNumberOfUnFeasible_i < numberOfUnFeasible_i AND
     ↪ newNumberOfUnFeasible_j < numberOfUnFeasible_j
    End
    return True
End Function
```

Anexo C. Demostraciones Capitulo 4

C.1. Demostración Proposición 4.6

En este caso tenemos que la complejidad para eliminar en vez de n^2 es $O(m\alpha(n,m))$, por simplicidad definimos $h = \max_{i\in[3,\dots,n],j\in[1,\dots,i(i-1)/2]} \alpha(i,j)$ de esta manera podemos considerar que eliminar puntos es $O(m)$ si es que al terminar los calculos se tiene la consideración de agregar h donde corresponda. También es importante recordar que $n_{clusters} \leq n$.

Primero miraremos el CVRP, la la cantidad de operaciones en los precálculos ahora están dadas por

$$\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} n_j + \sum_{i=1}^{n_{clusters}} \sqrt{n_i^3 \log(n_i)} + \sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j - 1)$$
$$+ \sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j - 1) + \sum_{i=1}^{n_{clusters}} \sqrt{(n_i-1)^3 \log(n_i-1)} + \sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j - 2)$$

notando que $\sqrt{n_i^3} = (n_i)^{\frac{3}{2}}$ podemos realizar las mismas cotas utilizadas en las partes anteriores y así obtenemos que

$$\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} n_j + \sum_{i=1}^{n_{clusters}} \sqrt{n_i^3 \log(n_i)} + \sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j - 1)$$
$$+ \sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j - 1) + \sum_{i=1}^{n_{clusters}} \sqrt{(n_i-1)^3 \log(n_i-1)} + \sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_j - 2)$$
$$\leq n(n + n_{clusters}) + \sqrt{(n+n_{clusters})^3 \log(n+n_{clusters})} + 2n^2 + \sqrt{n^3\log(n)} + (n - n_{clusters})n$$
$$\leq 5n^2 + \sqrt{(n+n_{clusters})^3 \log(n+n_{clusters})} + \sqrt{n^3\log(n)}$$

Ahora miraremos el Relocate, si denotamos por $\hat{i}, \hat{j}$ los clusters modificados tenemos que actualizar sus valores con lo que obtenemos

$$\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} \log(n_{clusters}) + \sum_{\substack{i=1 \\ i\neq \hat{i}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{i}} - 1) + \sum_{\substack{i=1 \\ i\neq \hat{j}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{j}} + 1)$$
$$+ \sqrt{(n_{\hat{i}}-1)^3 \log(n_{\hat{i}}-1)} + \sqrt{(n_{\hat{j}}+1)^3 \log\Big(n_{\hat{j}}+1\Big)} + \sum_{\substack{i=1 \\ i\neq \hat{i}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{i}} - 2)$$
$$+ \sum_{\substack{i=1 \\ i\neq \hat{j}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{j}}) + \sqrt{(n_{\hat{i}}-2)^3 \log(n_{\hat{i}}-2)} + \sqrt{(n_{\hat{j}})^3 \log\Big(n_{\hat{j}}\Big)}$$

con lo cual podemos acotar y obtener los siguiente

$$\sum_{i=1}^{n_{clusters}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{clusters}} \sum_{l=1}^{n_i-1} \log(n_{clusters}) + \sum_{\substack{i=1 \\ i\neq \hat{i}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{i}} - 1) + \sum_{\substack{i=1 \\ i\neq \hat{j}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{j}} + 1)$$
$$+ \sqrt{(n_{\hat{i}} - 1)^3 \log(n_{\hat{i}} - 1)} + \sqrt{(n_{\hat{j}} + 1)^3 \log\Big(n_{\hat{j}} + 1\Big)} + \sum_{\substack{i=1 \\ i\neq \hat{i}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{i}} - 2)$$
$$+ \sum_{\substack{i=1 \\ i\neq \hat{j}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{j}}) + \sqrt{(n_{\hat{i}} - 2)^3 \log(n_{\hat{i}} - 2)} + \sqrt{(n_{\hat{j}})^3 \log\Big(n_{\hat{j}}\Big)}$$
$$\leq (n_{clusters} - 1) n \log(n_{clusters}) + n(n_{\hat{i}} + n_{\hat{j}}) + \sqrt{(n_{\hat{i}} - 1)^3 \log(n_{\hat{i}} - 1)} + \sqrt{(n_{\hat{j}} + 1)^3 \log\Big(n_{\hat{j}} + 1\Big)}$$
$$+ n(n_{\hat{i}} + n_{\hat{j}} - 2) + \sqrt{(n_{\hat{i}} - 2)^3 \log(n_{\hat{i}} - 2)} + \sqrt{(n_{\hat{j}})^3 \log\Big(n_{\hat{j}}\Big)}$$
$$\leq (n_{clusters} - 1) n \log(n_{clusters}) + n(n + n_{clusters}) + \sqrt{n^3 \log(n)} + \sqrt{(n + 2n_{clusters})^3 \log(n + 2n_{clusters})}$$
$$+ n^2 + \sqrt{(n - n_{clusters})^3 \log(n - n_{clusters})} + \sqrt{(n + 2n_{clusters})^3 \log(n + 2n_{clusters})}$$

Por otro lado para Exchange obtenemos

$$\sum_{i=1}^{n_{clusters}} \sum_{j=i+1}^{n_{clusters}} \sum_{l=1}^{n_i-1} \sum_{m=1}^{n_j-1} \log(n_{clusters}) + \sum_{\substack{i=1 \\ i\neq \hat{i}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} n_{\hat{i}} + \sum_{\substack{i=1 \\ i\neq \hat{j}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} n_{\hat{j}} + \sqrt{n_{\hat{i}}^3 \log(n_{\hat{i}})}$$
$$+ \sqrt{n_{\hat{j}}^3 \log\Big(n_{\hat{j}}\Big)} + \sum_{\substack{i=1 \\ i\neq \hat{i}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{i}} - 1) + \sum_{\substack{i=1 \\ i\neq \hat{j}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{j}} - 1)$$
$$+ \sqrt{(n_{\hat{i}} - 1)^3 \log(n_{\hat{i}} - 1)} + \sqrt{(n_{\hat{j}} - 1)^3 \log\Big(n_{\hat{j}} - 1\Big)}$$

con esto obtenemos

$$\sum_{i=1}^{n_{clusters}} \sum_{j=i+1}^{n_{clusters}} \sum_{l=1}^{n_i-1} \sum_{m=1}^{n_j-1} \log(n_{clusters}) + \sum_{\substack{i=1 \\ i\neq \hat{i}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} n_{\hat{i}} + \sum_{\substack{i=1 \\ i\neq \hat{j}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} n_{\hat{j}} + \sqrt{n_{\hat{i}}^3 \log(n_{\hat{i}})}$$
$$+ \sqrt{n_{\hat{j}}^3 \log\Big(n_{\hat{j}}\Big)} + \sum_{\substack{i=1 \\ i\neq \hat{i}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{i}} - 1) + \sum_{\substack{i=1 \\ i\neq \hat{j}}}^{n_{clusters}} \sum_{l=1}^{n_i-1} (n_{\hat{j}} - 1)$$
$$+ \sqrt{(n_{\hat{i}} - 1)^3 \log(n_{\hat{i}} - 1)} + \sqrt{(n_{\hat{j}} - 1)^3 \log\Big(n_{\hat{j}} - 1\Big)}$$
$$\leq n^2 \log(n_{clusters}) + n(n + n_{clusters}) + \sqrt{(n + n_{clusters})^3 \log(n + n_{clusters})} + n^2 + \sqrt{n^3 \log(n)}$$

así obtenemos en total

$$5n^2 + h\sqrt{(n + n_{clusters})^3 \log(n + n_{clusters})} + h\sqrt{n^3 \log(n)}$$
$$+ T\Big[n^2 \log(n_{clusters}) + (n_{clusters} - 1)n \log(n_{clusters}) + 2n(n + n_{clusters})$$
$$+ 2h\sqrt{(n + n_{clusters})^3 \log(n + n_{clusters})} + 2n^2 + 2h\sqrt{n^3 \log(n)}$$
$$+ h\sqrt{(n - n_{clusters})^3 \log(n - n_{clusters})} + h\sqrt{(n + 2n_{clusters})^3 \log(n + 2n_{clusters})}\Big]$$

es decir

$$O\Big(n^2 + h\sqrt{(n + n_{clusters})^3 \log(n + n_{clusters})} + T\Big[n^2 \log(n_{clusters})$$
$$+ h\sqrt{(n + n_{clusters})^3 \log(n + n_{clusters})}\Big]\Big)$$

De manera similar para el VRPTW se concluye que:

Los precalculos tienen como cota de cantidad de operaciones a:

$$n(n + n_{clusters}) + \sqrt{(n + n_{clusters})^3 \log(n + n_{clusters})} + n^2 + \log(n)n^2 + n \log(n)$$
$$+ (n - n_{clusters})n \log(n - n_{clusters})$$

Relocate tiene como cota:

$$(n_{clusters} - 1)n \log(n_{clusters}) + n(n + n_{clusters}) + \sqrt{(n + 2n_{clusters})^3 \log(n + 2n_{clusters})}$$
$$+ \sqrt{n^3 \log(n)} + 2n^2 \log(n)$$

Mientras que Exchange:

$$n^2 \log(n_{clusters}) + n(n + n_{clusters}) + \sqrt{(n + n_{clusters})^3 \log(n + n_{clusters})} + 2n^2 \log(n)$$

con lo que en total tenemos

$$n(n + n_{clusters}) + h\sqrt{(n + n_{clusters})^3 \log(n + n_{clusters})} + n^2 + \log(n)n^2 + n \log(n)$$
$$+ (n - n_{clusters})n \log(n - n_{clusters}) + T\Big[n^2 \log(n_{clusters}) + (n_{clusters} - 1)n \log(n_{clusters})$$
$$+ 2n(n + n_{clusters}) + h\sqrt{(n + n_{clusters})^3 \log(n + n_{clusters})}$$
$$+ h\sqrt{(n + 2n_{clusters})^3 \log(n + 2n_{clusters})} + h\sqrt{n^3 \log(n)}4n^2 \log(n)\Big]$$

luego utilizando $n_{clusters} \leq n$ se obtiene que

$$O\Bigg(n^2 \log(n) + h\sqrt{(n + n_{clusters})^3 \log(n + n_{clusters})} \\ + T\left[n^2 \log(n) + h\sqrt{(n + n_{clusters})^3 \log(n + n_{clusters})}\right]\Bigg)$$

Anexo D. Función Objetivo y Tiempos De Ejecución

En los próximos gráficos se presenta el efecto que tiene α sobre la función objetivo y los tiempos de ejecución.

D.1. CVRP

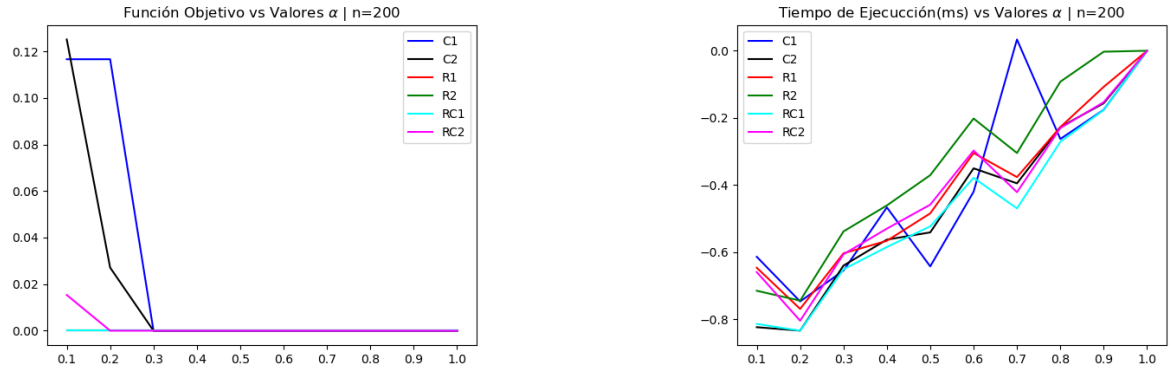


Figura D.1: Gráficos instancias tamaño 200

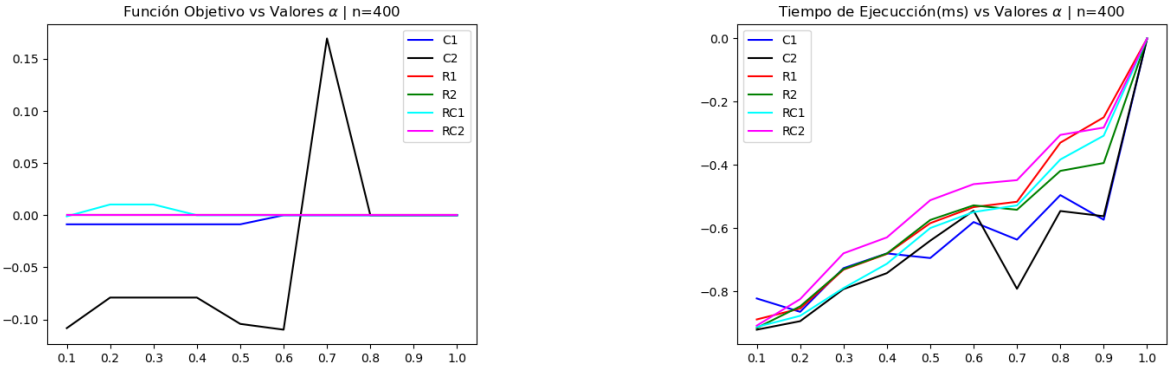


Figura D.2: Gráficos instancias tamaño 400

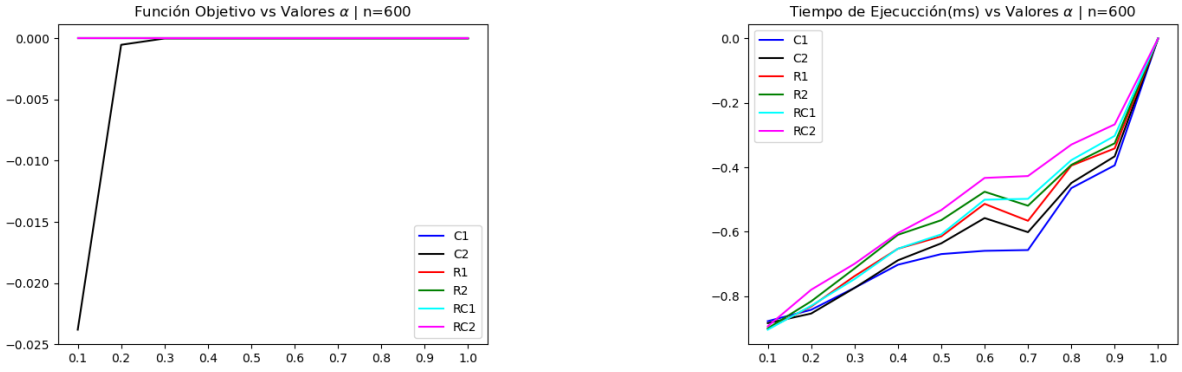


Figura D.3: Gráficos instancias tamaño 600

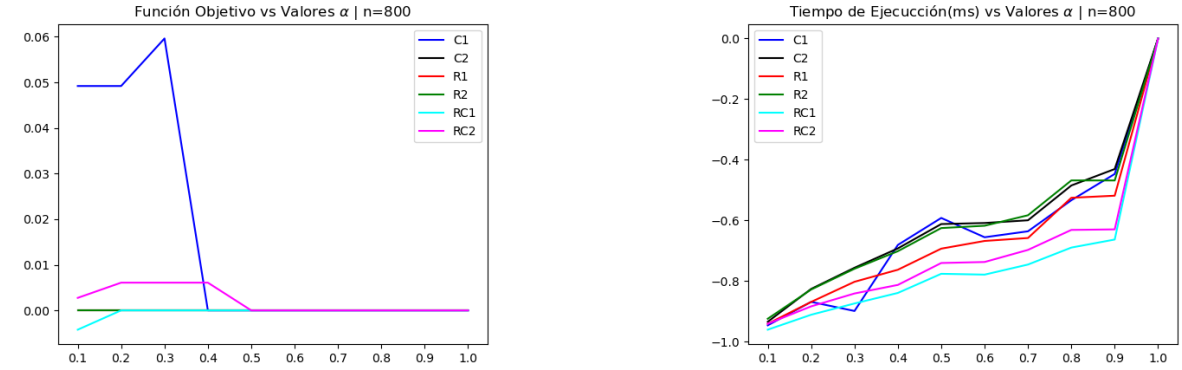


Figura D.4: Gráficos instancias tamaño 800

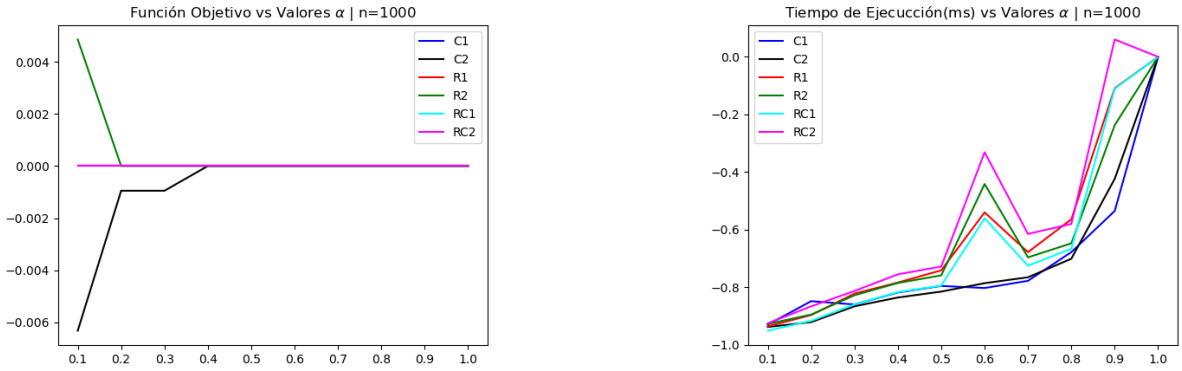


Figura D.5: Gráficos instancias tamaño 1000

D.2. VRPTW

Dada la gran cantidad de casos para el VRPTW no es posible identificar cada una de las instancias de manera individual y solo se hará de manera macro con el tipo $C1$, $C2$, $R1$, $R2$, $RC1$, $RC2$ y el tamaño $200, 400, 600, 800, 100$.

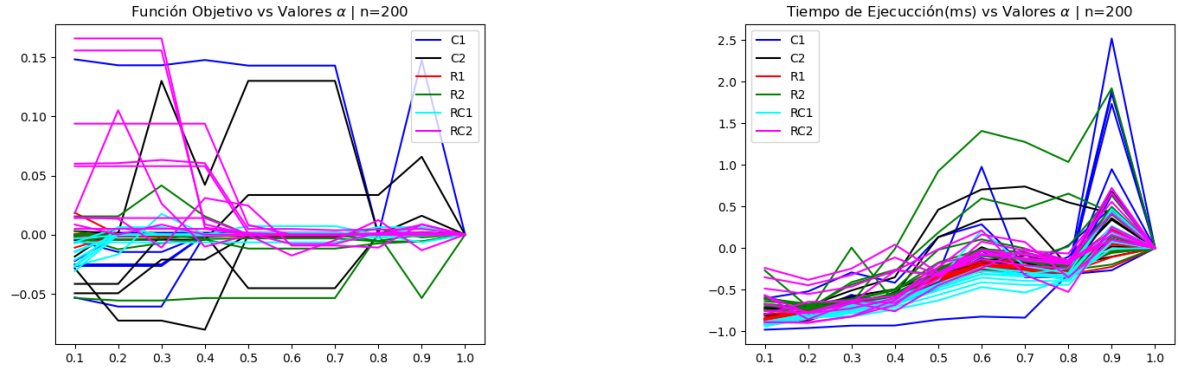


Figura D.6: Gráficos instancias tamaño 200

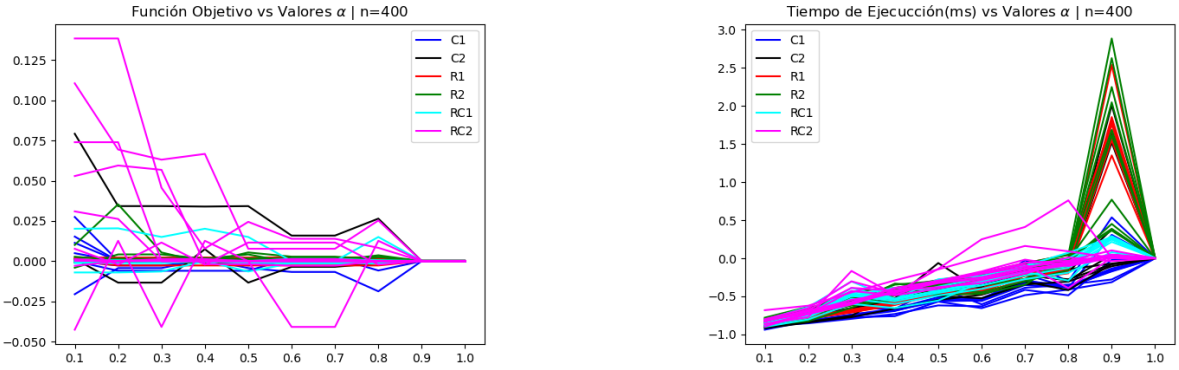


Figura D.7: Gráficos instancias tamaño 400

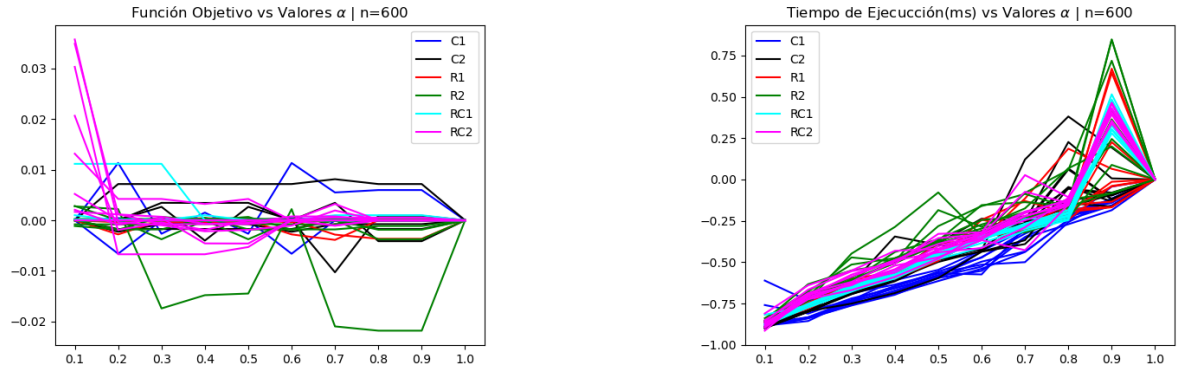
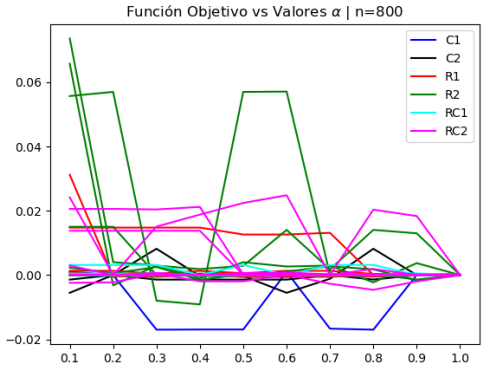


Figura D.8: Gráficos instancias tamaño 600



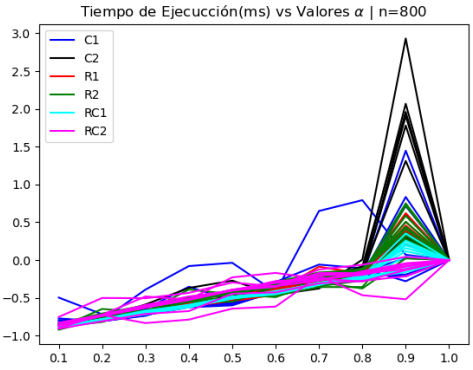
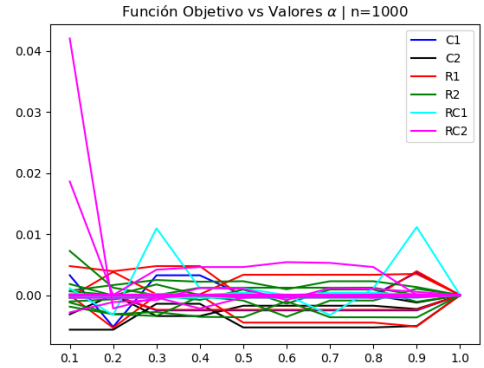


Figura D.9: Gráficos instancias tamaño 800



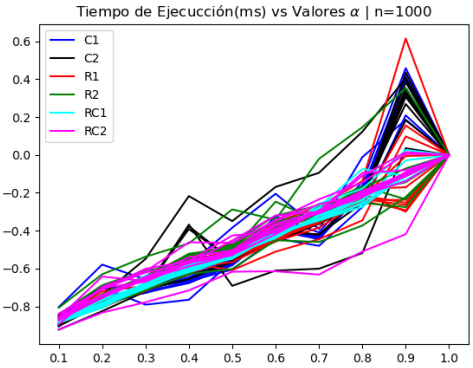


Figura D.10: Gráficos instancias tamaño 1000