UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

Peumo Repositorio Digital USM **https://repositorio.usm.cl**

Departamento de Informática Magíster en Ciencias de la Ingeniería Informática (MII)

2021-08

UN FRAMEWORK PARA LA CREACIÓN DE INSTANCIAS DEL PROBLEMA DE RUTAS DE TRÁNSITO URBANO

Díaz Urra, Roberto Nicolás

https://hdl.handle.net/11673/56476

Downloaded de Peumo Repositorio Digital USM, UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE INFORMÁTICA

UN FRAMEWORK PARA LA CREACIÓN DE INSTANCIAS DEL PROBLEMA DE RUTAS DE TRÁNSITO URBANO

Roberto Nicolás Díaz Urra

MAGÍSTER EN CIENCIAS DE LA INGENIERÍA INFORMÁTICA

AGOSTO 2021

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA

UN FRAMEWORK PARA LA CREACIÓN DE INSTANCIAS DEL PROBLEMA DE RUTAS DE TRÁNSITO URBANO

Tesis de Grado presentada por
ROBERTO NICOLÁS DÍAZ URRA

como requisito parcial para optar al grado de
MAGÍSTER EN CIENCIAS DE LA INGENIERÍA INFORMÁTICA

DIRECTOR DE TESIS
CARLOS CASTRO VALDEBENITO

CO-DIRECTOR DE TESIS
NICOLÁS GÁLVEZ RAMÍREZ

AGOSTO 2021

TÍTULO DE LA TESIS:

UN FRAMEWORK PARA LA CREACIÓN DE INSTANCIAS DEL PROBLEMA DE RUTAS DE TRÁNSITO URBANO

AUTOR:

ROBERTO NICOLÁS DÍAZ URRA

Trabajo de Tesis, presentado en cumplimiento parcial de los requisitos para el Grado de **Magíster en Ciencias de la Ingeniería Informática** de la Universidad Técnica Federico Santa María.

Carlos Castro _____
Director de Tesis

Nicolás Gálvez _____
Co-Director de Tesis

Claudio Torres _____
Correferente Interno

Broderick Crawford _____
Correferente Externo

Marcelo Mendoza _____
Presidente Comisión Evaluadora

Agosto 2021.

Santiago, Chile.

Agradecimientos

Me gustaría agradecer al profesor Carlos Castro por acoger este trabajo de tesis considerando la situación en que me encontraba. También agradecer al profesor Nicolás Gálvez por su invaluable apoyo y consejos durante todo el desarrollo de este trabajo de investigación. Ambos fueron una base sólida que brindó un sustento necesario para que este trabajo llegase a buen puerto.

Quiero también agradecer a mi familia pues su apoyo ha sido imprescindible en el desarrollo de mi persona y han tenido la comprensión necesaria para esperar el término de este trabajo.

Por último, pero no menos, agradecer a mis amigos y compañeros, que siempre estuvieron ahí para apoyarme y sacar este trabajo adelante.

Muchas gracias a todos.

Resumen

Los sistemas de transporte son componentes críticos para las ciudades, impactando inmensamente la calidad de vida de sus ciudadanos, proveyendo alternativas de transporte, reduciendo drásticamente el tráfico vehicular y la contaminación atmosférica. Se requiere de una cuidadosa planificación para evitar usuarios descontentos y un sistema insostenible, siendo fundamental el correcto diseño de la red de rutas de buses. En consecuencia, el Urban Transit Routing Problem (UTRP) se enfoca en encontrar un conjunto de rutas de buses que minimiza el tiempo de viaje de los pasajeros y los costos al operador del sistema. Varios algoritmos han sido desarrollados para resolver el UTRP, pero la mayoría de las instancias del problema carecen de datos de demanda de la vida real, con las instancias más conocidas siendo muy pequeñas para los estándares actuales y/o generadas aleatoriamente. Las técnicas de relajación del estado del arte se basan en características inherentes de los sistemas de transporte urbano y no pueden reducir de manera significativa el orden de magnitud de instancias complejas. En este trabajo, se propone un framework para generar instancias de UTRP usando datos de demanda zonal de la vida real, que incluyen miles de ubicaciones de paraderos. Los algoritmos de clustering permiten al framework reducir la complejidad del problema generando una aproximación que mantiene el comportamiento de la demanda y la estructura de caminos manteniendo conectadas y representadas tanto ubicaciones centrales como periféricas. El framework se aplica al complejo sistema de transporte público de la ciudad de Santiago de Chile. Se generan instancias con un comportamiento similar de demanda al usar una cantidad suficiente de clústeres. Además, los diversos algoritmos de clustering probados muestran una alta similitud en su salida y rendimiento. Este framework es fácilmente aplicable a diferentes realidades y debería ayudar a futuros investigadores en el diseño de algoritmos de resolución, así como mejorar los modelos de aproximación de otras ciudades.

Abstract

Transportation systems are critical components of urban cities, by hugely impacting the quality of life of their citizens, providing alternative means of transportation and drastically reducing street traffic and atmospheric pollution. Careful planning is required to avoid disgruntled users and an unsustainable system, where the correct design of the bus routes network is a fundamental task. Thus, Urban Transit Routing Problem (UTRP) aims to find a set of bus routes that minimises users travelling time and system operator costs. Several algorithms have been developed to solve the UTRP, but most problem instances lack real-life demand data information, where well-known benchmark instances are very small w.r.t. current standards and/or they have been randomly generated. State-of-the-art relaxation techniques are based on inherent features of urban transport systems and they cannot significantly reduce the order of magnitude of complex instances. In this work, we propose a framework to generate UTRP instances using real-life zone-based demand data, including thousand of bus stops locations. Clustering algorithms allow the framework to reduce the problem complexity by generating an approximation that preserves demand behaviour and road structure by keeping central and periphery locations represented and connected. Our study case is the complex public transport system of Santiago, Chile. We generate instances that have similar demand behaviour when using enough amount of clusters. Also, the use of several cluster algorithms shows similar performance and output. This framework is easily applicable to different realities and it should help future researchers to test network design algorithms as well as to improve the approximation models of other cities.

Glosario

- **Algoritmo**: Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema, llevar a cabo una tarea o realizar un cálculo.
- **Algoritmo de Clustering**: Algoritmo que distribuye un conjunto de objetos en varios subconjuntos mutuamente exclusivos que sean coherentes internamente, pero claramente diferentes de cada otro.
- **Clúster**: Subconjunto o grupo de objetos, similares entre sí, generado por un Algoritmo de Clustering.
- **Framework**: Conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.
- **Función objetivo**: Medida cuantitativa del funcionamiento del sistema que se desea optimizar.
- **Grafo**: Conjunto de nodos unidos por enlaces que representan relaciones entre ellos.
- **Histograma**: Representación aproximada de la distribución de datos númericos que permite apreciar la frecuencia de ciertos valores.
- **Instancia**: Versión del problema con ciertas características definidas.
- **Matriz**: Arreglo bidimensional de números dispuestos en filas y columnas, usado para representar un objeto matemático o una propiedad de tal clase de objeto.

- **Matriz de Demanda**: Matriz en el que cada entrada representa el número de viajes entre un par de ubicaciones en algún período de tiempo, existiendo una fila y una columna por cada ubicación de interés en la ciudad.
- **Matriz de Demanda Relajada**: Aproximación o simplificación de otra Matriz de Demanda. Obtenida al aplicar un mecanismo de reducción.
- **Matriz de Demanda Zonal Simple**: Matriz de Demanda en que cada fila y columna corresponde a una zona de la ciudad.
- **Matriz de Demanda Zonal Temporal**: Matriz de Demanda tridimensional, en que cada fila y columna representa una zona, y cada capa un período de tiempo.
- **Problema de Rutas de Tránsito Urbano**. Problema Multi-objetivo que consiste en encontrar un conjunto de rutas para una ciudad de forma que sus habitantes puedan transportarse rápida y cómoda a sus destinos, manteniendo un bajo costo para la manutención del sistema.
- **Problema Multi-objetivo**: Problema de optimización que tiene dos o más funciones objetivos asociadas.
- **RDM**: Del inglés *Relaxed Demand Matrix*, Matriz de Demanda Relajada.
- **Red de Caminos Relajada**. Grafo que representa una aproximación a los caminos de una ciudad. Obtenida al aplicar un mecanismo de reducción.
- **RRN**: Del inglés *Relaxed Road Network*, Red de Caminos Relajada.
- **UTRP**: Del inglés *Urban Transit Routing Problem*, Problema de Rutas de Tránsito Urbano.

Índice de Contenidos

1. Introducción **1**

1.1. Objetivos 3

1.1.1. Objetivo general 4

1.1.2. Objetivos específicos 4

1.2. Contribuciones 5

1.3. Estructura de la tesis 5

2. El Problema de Rutas de Tránsito Urbano **7**

2.1. Problemas de optimización combinatoria 7

2.2. Problemas multi-objetivo 11

2.3. Problemas relacionados 13

2.3.1. Problemas de diseño de red 14

2.3.2. Problemas de transporte 15

2.4. Hacia el UTRP 19

2.5. Formalización de UTRP 22

2.5.1. Elementos básicos 22

2.5.2. Evaluación de Soluciones 24

2.6. Análisis de Instancias previas del UTRP 28

3. Algoritmos de clustering y similitud **34**

3.1. Algoritmos de clustering 34

3.1.1. Clustering 34

3.1.2. K-Means 35

3.1.3. Clustering Jerárquico Aglomerativo 37

3.1.4. Clustering espectral 38

3.1.5. Clustering basado en modelo 40

3.1.6. Mean-Shift 42

3.1.7. Affinity Propagation 43

3.1.8. DBSCAN 44

3.1.9. OPTICS 46

3.2. Medidas de similitud 48

3.2.1. Índice Rand 49

3.2.2. Índice Rand Ajustado 50

3.2.3. Índice de Emparejamiento Único 52

4. Framework para la creación de instancias **55**

4.1. Red de Caminos Relajada 57

4.2. Matriz de Demanda Relajada 58

4.2.1. Peso basado en Ubicación 59

4.2.2. Peso basado en Distancia 59

4.2.3. Factor de Doble Peso 60

4.2.4. Reducción de dimensión temporal en datos de demanda 62

4.2.5. Agregación de resultados 63

4.3. Parámetros de las rutas 63

5. Experimentación **65**

5.1. Configuración experimental 65

5.1.1. Algoritmos de Clustering 65

5.1.2. Red de caminos relajada 66

5.1.3. Matriz de demanda relajada 67

5.1.4. Parámetros de las rutas 68

5.1.5. Ambiente de experimentación 68

5.2. Resultados y Discusión 69

5.2.1. Recomendación de algoritmos de clustering 70

5.2.2. Instancias generadas 71

5.2.3. Distribución de demanda 74

5.2.4. Similitud entre algoritmos de clustering 78

6. Conclusiones y Perspectivas **80**

6.1. Perspectivas 81

A. Resultados completos **83**

A.1. Redes de Caminos Relajadas 83

A.2. Comportamiento de la Demanda 86

A.3. Histogramas de demanda 88

A.4. Similitud entre algoritmos de clustering 90

Bibliografía **93**

Lista de Tablas

2.1. Clasificación de problemas de búsqueda 9

2.2. Resumen de instancias UTRP. 29

2.3. Generación de elementos básicos de las instancias UTRP 31

5.1. Características de las instancias UTRP generadas 72

5.2. ARI entre algoritmos de clustering 79

5.3. UMI entre algoritmos de clustering 79

A.1. ARI entre algoritmos de clustering (35 clústeres) 90

A.2. ARI entre algoritmos de clustering (90 clústeres) 91

A.3. ARI entre algoritmos de clustering (135 clústeres) 91

A.4. UMI entre algoritmos de clustering (35 clústeres) 91

A.5. UMI entre algoritmos de clustering (90 clústeres) 92

A.6. UMI entre algoritmos de clustering (135 clústeres) 92

Lista de Figuras

2.1. Vista de caja negra para definición de problema 8

2.2. Ejemplo gráfico de hipervolumen 14

3.1. Ejemplo de Algoritmo OPTICS 48

3.2. Ejemplo de Índice de Emparejamiento Único 54

4.1. Diagrama de flujo de datos del framework de creación de instancias. 55

4.2. Ejemplo de construcción de RRN 58

4.3. Ejemplo de Factor de doble peso 61

5.1. Diagrama de flujo de datos del framework de creación de instancias aplicado a los datos de RED. 69

5.2. Comparación de RRN generada por diferentes algoritmos de clustering. . . 71

5.3. Relajación de la instancia RED al aplicar clustering basado en Mezcla de Gaussianas. 72

5.4. Comparación de la estructura de la red de caminos 73

5.5. Comparación de la distribución de demanda 75

5.6. Histograma de demanda de instancia Mandl 76

5.7. Histogramas de matrices de demanda de instancias de Mumford 76

5.8. Histograma de demanda de instancias Nottingham100 y Edinburgh200 . . . 77

5.9. Comparación de demanda entre Clustering Aglomerativo y Matriz de demanda zonal simple 78

A.1. Red Aproximada de Caminos 84

A.2. RRN de instancias generadas con K-Means 84

A.3. RRN de instancias generadas con Clustering Aglomerativo 85

A.4. RRN de instancias generadas con Clustering basado en mezcla de Gaussianas 85

A.5. RRN de instancias generadas con Clustering Espectral 85

A.6. Distribución de demanda de las zonas de RED 86

A.7. Comportamiento de demanda al usar K-Means 86

A.8. Comportamiento de demanda al usar Clustering Aglomerativo 87

A.9. Comportamiento de demanda al usar Clustering basado en mezcla de Gaussianas 87

A.10. Comportamiento de demanda al usar Clustering Espectral 88

A.11. Comparación de demanda entre K-Means y Matriz de demanda zonal simple 88

A.12. Comparación de demanda entre Clustering Aglomerativo y Matriz de demanda zonal simple 89

A.13. Comparación de demanda entre Mezcla de Gaussianas y Matriz de demanda zonal simple 89

A.14. Comparación de demanda entre Clustering Espectral y Matriz de demanda zonal simple 90

Lista de Definiciones

2.1. Red de Caminos 23

2.2. Matriz de tiempos de viaje 23

2.3. Matriz de demanda 23

2.4. Parámetros de las rutas 24

2.5. Red de Rutas 24

2.6. Función Objetivo de Tiempo de Viaje Promedio 25

2.7. Función Objetivo de Costo al operador 26

2.8. Restricciones UTRP 27

3.1. Centroide 35

3.2. Clústeres 35

3.3. Índice Rand 49

3.4. Índice Rand Ajustado 50

3.5. Índice de Emparejamiento Único 52

4.1. Zonas 56

4.2. Red de Caminos Relajada 57

4.3. Peso basado en Ubicación 59

4.4. Peso basado en Distancia 59

4.5. Factor de Doble Peso 60

4.6. Matriz de Demanda Zonal Simple 62

4.7. Matriz de Demanda Relajada 63

5.1. Transformación de coordenadas geográficas 67

Capítulo 1

Introducción

Los sistemas de transporte urbano forman parte esencial de una gran ciudad. Al generar alternativas al automóvil, reducen su uso para transportarse desde un punto a otro, beneficiando a las personas sin poder adquisitivo, descongestionando las calles y reduciendo la contaminación atmosférica producto de su excesivo uso [1, 2].

Estos sistemas plantean un gran desafío: deben ser planeados cuidadosamente para ser capaces de llevar a las personas de forma rápida, efectiva y cómoda a sus destinos. Un sistema mal diseñado genera insatisfacción y descontento en la población, afectando negativamente la calidad de vida de los ciudadanos [3].

Se debe considerar que el mantenimiento de un sistema de transporte tiene un costo asociado. Compra y mantención de buses, combustible, choferes y toda la logística necesaria para coordinar al sistema. De esta forma, no planificar adecuadamente puede incurrir en un costo muy elevado y hacer que el sistema no sea económicamente sustentable.

Por lo tanto, una ciudad necesita contar con un sistema de transporte cuyos recorridos sean capaces de satisfacer la demanda de la población y llevarla a su destino a tiempo, tanto como no incurrir en costos muy elevados para el operador del servicio.

El proceso de planificación que debería llevarse a cabo antes de la implementación del sistema de transporte es conocido como el **Problema de Diseño de Red de Tránsito urbano**

(UTNDP del inglés *Urban Transit Network Design Problem*) [4, 5] y usualmente se divide en varios sub-problemas o fases que involucran decisiones tácticas, estratégicas y operacionales.

En [6] se indica que se pueden considerar cinco fases en el diseño de un sistema de transporte de buses: diseño de la red, ajuste de frecuencias, desarrollo de horarios, itinerario de buses e itinerario de conductores. Cada una de éstas es un problema complejo por sí solo, por lo que usualmente no se resuelven todas las fases al mismo tiempo a pesar de su interdependencia.

Una formulación que contempla la primera de estas fases, y por tanto, parte fundamental del diseño de un sistema de transporte [5] se conoce como el **Problema de Rutas de Tránsito Urbano** (UTRP del inglés *Urban Transit Routing Problem*) que consiste en la creación de rutas para el transporte público. Se trata de un problema *NP-Hard* [7, 8] altamente complejo.

En [5] se expone que para el UTRP aún existen diferentes modelos que pueden variar en objetivos y restricciones. En esta tesis nos enfocamos en la versión propuesta en [9], que ha sido utilizada en diversos trabajos como [4], [10], [11], [8], [2], [12], [13] y [14].

Diversas técnicas han sido utilizadas para abordar el UTRP, basadas principalmente en metaheurísticas. Los algoritmos genéticos se encuentran entre las técnicas más usadas [8, 11, 9, 4, 12], pero también se han utilizados algoritmos de búsqueda local [10], optimización por enjambre de partículas [2] e hiperheurísticas [14].

Un desafío que se enfrenta al resolver este problema es la carencia de datos para la comparación de métodos de resolución. Se pueden identificar dos causas [8]: (1) La colección de datos respecto de transporte público es difícil de obtener, porque los pasajeros no pueden ser fácilmente rastreados y las encuestas pueden ser bien costosas e imprecisas, y (2) que para la evaluación y calidad de soluciones en sistemas de la vida real se ha hecho uso de métricas ad-hoc sumamente diversas, así que solo pueden ser usadas en pocos escenarios.

La mayoría de las instancias que es posible encontrar para el UTRP son muy pequeñas, generadas aleatoriamente [8] y/o basadas en suposiciones de la densidad de población [15].

Además, incluso las más grandes instancias dentro de la literatura son muy pequeñas respecto de escenarios de la vida real. Para que las técnicas de resolución sean capaces de abordar

el problema se necesitan aproximaciones ajustadas a estos complejos casos, relajando la cantidad de paraderos y manteniendo una distribución de demanda similar.

Una reducción básica puede ser encontrada en [16], donde las instancias del UTRP son relajadas combinando nodos puestos entre las esquinas de las calles. Esta compresión es similar a una característica inherente en la mayoría de los sistemas de transporte urbano: varios paraderos se esparcen en grandes áreas debido a la dirección del tráfico o por la convergencia del público, pero pueden ser vistas como la misma ubicación estratégica. Sin embargo, esta relajación es insuficiente para escenarios complejos.

Un escenario complejo bien conocido es el sistema de transporte urbano de Santiago de Chile, la llamada *Red Metropolitana de Movilidad* (*RED*), anteriormente conocida como Transantiago. Este sistema obtuvo atención mundial en el año 2007 después de su prematura implementación a pesar de varios problemas de diseño, e.g., falta de infraestructura crítica y condiciones de sistema aún no puestas en operación [17]. Hoy día, el sistema de buses de *RED* transporta cerca de 6,2 millones de usuarios de 34 comunas del área metropolitana de Santiago. Cubre cerca de 680 km^2 en zona urbanas [18] y comprende 11.320 paraderos agrupados en más de 800 zonas de demanda.

1.1. Objetivos

Tomando en consideración lo anterior: (1) la falta de instancias para el UTRP que aproximen adecuadamente casos reales y complejos, (2) la dificultad de resolver el UTRP para casos complejos con miles de paraderos y (3) la falta de métodos de reducción de instancias más generales se plantea la siguiente hipótesis:

El uso de una técnica de clustering genera una instancia para el UTRP de buena calidad, esto es, entregando una red de caminos que mantiene características de la red original y patrones de demanda de viaje similares al caso real

Las técnicas de clustering permiten generar una instancia con un tamaño órdenes de magnitud menor que una traducción directa de los datos del escenario real. Sin embargo, en esta

reducción se debe mantener una estructura similar en la red de caminos y que los patrones de demanda sean similares para que sea una versión válida y tenga el potencial de apoyar el diseño de un sistema de transporte de buses.

Las componentes de la instancia, esto es la red de caminos y la matriz de demanda, debe aproximar adecuadamente el escenario original. La red de caminos debe mantener características estructurales evitando crear caminos a través de fallas geográficas que no lo permiten. La matriz de demanda debe mantener los mismos patrones de comportamiento de los usuarios de la red original.

Las instancias generadas con este método, basadas en datos de origen-destino, podrían tener características que las diferenciaran significativamente de algunas instancias de la literatura como las que fueron generadas casi aleatoriamente [8] y/o basadas en suposiciones de densidad de la población [15].

1.1.1. Objetivo general

Para la comprobación de la hipótesis planteada, se define el siguiente objetivo general:

Diseñar e implementar un framework que permita la generación de instancias para el Problema de Rutas de Tránsito Urbano que sean una aproximación para una gran ciudad.

1.1.2. Objetivos específicos

Para cumplir con el objetivo general se definen los siguientes objetivos específicos:

- Analizar los avances en la resolución del UTRP y problemas similares, enfocándose en la representación de un sistema de transporte urbano y la generación de instancias.
- Diseñar un framework para la generación de instancias a partir de datos geográficos de paraderos.
- Generar nuevas instancias a partir de datos de origen-destino de Santiago de Chile.

- Evaluar y analizar críticamente el framework comparando con técnicas anteriores de generación de instancias y la calidad de su salida como una aproximación de los datos reales.

1.2. Contribuciones

Las contribuciones de este trabajo de investigación son las siguientes:

- Una técnica que permite generar instancias que son una aproximación de una red de tránsito de tamaño real.
- Una nueva instancia basada en datos de origen-destino de Santiago de Chile.

1.3. Estructura de la tesis

Lo que resta de esta tesis está estructurado de la siguiente forma:

- **Capítulo 2 - El Problema de Rutas de Tránsito Urbano**: Se revisitan conceptos necesarios que fundamentan este trabajo, cómo se aborda el UTRP y la formalización de este problema. Además, se investiga los avances históricos y científicos del UTRP, se mencionan problemas relacionados y se revisan las técnicas utilizadas para la generación de instancias.
- **Capítulo 3 - Algoritmos de clustering y similitud**: Se revisa este tipo de técnicas y conceptos atingentes. Además de ofrecer una revisión de los distintos algoritmos de clustering, se presentan formas de medir la similitud de la salida producida por este tipo de técnicas para realizar análisis numéricos comparativos.
- **Capítulo 4 - Framework para la creación de instancias**: Descripción detallada del framework desarrollado para la generación de instancias reducidas.

- **Capítulo 5 - Experimentación**: Se presenta la configuración experimental y un análisis crítico de los resultados obtenidos a partir de la técnica.
- **Capítulo 6 - Conclusiones y Perspectivas**: Se encuentran las principales conclusiones derivadas del trabajo realizado y los resultados experimentales. Se proponen futuras líneas de investigación que surgen a partir de esta investigación.

Capítulo 2

El Problema de Rutas de Tránsito Urbano

En este capítulo se presenta el Problema de Rutas de Tránsito urbano o UTRP con análisis que va desde lo general a lo específico: se parte describiendo los problemas de optimización combinatoria, para seguir con problemas multi-objetivo. Posteriormente se presentan los problemas de diseño de red y problemas de transporte, super-problemas que abarcan al UTRP. El capítulo termina presentando como surge el UTRP, trabajos que se enfocan en su resolución, una formalización de éste y sus componentes, y un análisis de las instancias existentes para el problema.

2.1. Problemas de optimización combinatoria

Un problema, tal como se consigna en [19], puede ser visto como una caja negra que consiste en un cierto modelo como lo ilustra la Figura 2.1. Este modelo recibe cierta entrada y, a partir de esta, entrega cierta salida.

El tipo de problema dependerá de cuál de estas tres componentes sea conocida o especificada. Podemos clasificar los problemas en tres tipos:

- **Optimización**: Donde el modelo es conocido. La salida es conocida (e.g. la función deba alcanzar el valor cero) o solo descrita (e.g. la función debe alcanzar el valor mínimo o máximo posible). La tarea es encontrar la entrada que lleva a esta salida.
- **Modelado**: En este caso, la entrada y la salida son conocidas. Lo que se quiere conocer es el modelo que permite obtener cada salida para cada entrada. La gracia de contar con este modelo es que debería permitir realizar predicciones sobre entradas aún no conocidas.
- **Simulación**: Dado un modelo y algunas entradas, la tarea es calcular la salida correspondiente. De esta forma se pueden realizar, para un cierto sistema, pronósticos a futuro o predecir su comportamiento con cierta configuración.

Aunque la resolución de un problema de simulación puede ser visto simplemente como aplicar el modelo a la entrada para obtener la correspondiente salida, para los problemas de optimización y modelado la tarea será la búsqueda de una cierta configuración en un espacio de posibilidades, es decir, un problema de búsqueda.

Un problema de búsqueda tendrá asociado la noción de **espacio de búsqueda**, conjunto de todas las combinaciones de valores para las variables, esto es, todas las soluciones candidatas del problema. La tarea será encontrar, evaluar y aceptar estas soluciones candidatas, y obtener aquella que cumpla con el criterio establecido.

En el caso de los problemas de optimización, la solución va a corresponder a aquellos valores de entrada que, al aplicar el modelo, entregan una cierta salida especificada de forma explícita o implícita. En el caso de los problemas de modelado se quiere el modelo que entregue la mayor cantidad de veces la salida correcta para cada entrada.

Para la evaluación de cada solución se hará uso de una función específica conocida como la **función objetivo**. La función objetivo será una medida de la calidad de la solución. La

Figura 2.1: Vista de caja negra para definición de problema

aceptación de una solución dependerá de si ésta satisface el **conjunto de restricciones** del problema. Una solución candidata será considerada como una **solución factible** si es que satisface las restricciones del problema.

Por tanto, los problemas de búsqueda pueden ser clasificados, como lo muestra la Tabla 2.1, de acuerdo a la presencia o no de la función objetivo y del conjunto de restricciones (que guiarán la búsqueda).

Restricciones	Función Objetivo	
	Sí	No
Sí	Problema de Optimización con Restricciones (CSOP)	Problema de Satisfacción de Restricciones (CSP)
No	Problema de Optimización Libre (FOP)	No hay problema

Tabla 2.1: Clasificación de problemas de búsqueda

Por tanto, si solo la función objetivo está presente se tiene un Problema de Optimización Libre (FOP, del inglés *Free Optimization Problem*). Si solo se tiene un conjunto de restricciones a satisfacer se tiene un Problema de Satisfacción de Restricciones (CSP, del inglés *Constraint Satisfaction Problem*). Si ambos elementos están presentes se tiene un Problema de Optimización con Restricciones (CSOP, del inglés *Constraint Satisfaction Optimization Problem*).

Un problema de optimización combinatoria es un problema de búsqueda que consiste en encontrar una solución desde un espacio de búsqueda finito (posiblemente infinito contable). La solución está descrita por variables con valores discretos y puede tomar varias formas [20]: un entero, un subconjunto, una permutación, una estructura de grafo, etc.

Un problema de optimización combinatoria (que es además un problema de optimización con restricciones) $P = (E, f)$ puede ser definido por:

- Un conjunto de variables $X = \{x_1, ..., x_n\}$

- Dominios de las variables $D_1, ..., D_n$
- Restricciones entre las variables
- Una función objetivo f a ser minimizada (o maximizada) donde $f : D_1 \times ... \times D_n \rightarrow \mathbb{R}^+$

El espacio de búsqueda E de todas las posibles asignaciones (o de soluciones candidatas) queda definido por:

$$E = \{s = \{(x_1, v_1), ..., (x_n, v_n)\} \mid v_i \in D_i, s \text{ satisface las restricciones } \}$$

Siendo v_i el valor asignado a la variable x_i correspondiente dentro de una solución del conjunto.

Para resolver el problema de optimización combinatoria se debe encontrar la solución $s^* \in E$ con valor óptimo de función objetivo, es decir, si $f(s^*) < f(s)\ \forall s \in E$ entonces s^* es la solución global o el óptimo global del problema $P = (E, f)$ [20].

Si una solución satisface todas las restricciones se considera una **solución factible** y es una solución candidata al problema. En caso de que una solución no satisfaga al menos una de las restricciones se considerará una **solución infactible** y se considerará inválida. No obstante, las soluciones infactibles son usadas por algunos acercamientos pues limitar la búsqueda tomando en cuenta todas las restricciones puede llevar a soluciones mediocres [21, sección 2.4.3]. Existen técnicas para manejar la no satisfacción de las restricciones, por ejemplo, usar una función de penalización [22] [23, sección 2.4] o alternar entre fases de optimización y reparación [24] .

Un ejemplo sencillo de problema de optimización combinatoria es el Problema de la Mochila (o en inglés *Knapsack Problem*) que consiste en elegir entre un conjunto de objetos de forma de maximizar el valor o beneficio que estos otorgan, pero sin sobrepasar un límite de peso. Este problema se puede formalizar como se muestra en el Ejemplo 2.1.

Ejemplo 2.1 (Problema de la Mochila). Sea n el número de objetos, b_i y w_i el beneficio y el peso asociado a cada objeto con $i \in \{1, 2, \ldots, n\}$, respectivamente. Sea W el peso máximo. El **Problema de la Mochila** $P = (E, f)$ se puede definir como:

- **Variables:** $o_i \; \forall i \in \{1, 2, \ldots, n\}\}$
- **Dominios:** $o_i \in \{0, 1\}$
- **Función objetivo:**
$$\text{Max: } f = \sum_{i=1}^{n} b_i o_i$$
- **Restricciones:**
$$\sum_{i=1}^{n} w_i o_i \leq W$$

Cada variable o_i representa si un cierto objeto es llevado en la mochila o no (1 si se lleva, 0 si no). La función objetivo busca la maximización del beneficio lo cual es un producto simple de las variables o_i que se llevan con su beneficio b_i. Solo aportaran al valor de la sumatoria aquellos objetos tales que $o_i = 1$. La única restricción del problema, que asegura que la suma de los objetos llevados no supere el peso máximo W, presenta una estructura similar, esta vez multiplicando por el peso w_i en lugar del beneficio.

2.2. Problemas multi-objetivo

En la sección anterior se presentaron los problemas de optimización combinatoria teniendo un único criterio de optimización o evaluación. Es normal que aparezcan problemas con múltiples objetivos los que pueden tener conflictos entre sí, es decir, que cuando uno mejore el otro, normalmente, empeore.

Un **problema de optimización combinatoria multi-objetivo** es aquel que en lugar de tener una sola función objetivo f tiene un conjunto $F = \{f_1, ..., f_n\}$ de funciones objetivo donde el objetivo es minimizar (o maximizar) cada una de las funciones objetivos [25].

Hay un cambio en la noción de óptimo al haber múltiples objetivos. En optimización multi-objetivo la meta cambia a encontrar *trade-offs* más que una mejor solución individual como en mono-objetivo. Los múltiples objetivos usualmente se contraponen entre sí, la reducción de valor en uno conlleva un aumento en otro. Por tanto, en multi-objetivo se desea un conjunto óptimo de soluciones de las cuales escoger [23].

En un contexto multi-objetivo para la comparación de soluciones, a menudo se aplica el concepto de **dominancia de Pareto** [25] [23]. Si se tienen dos soluciones s_1, s_2 se dirá que s_1 domina s_2 (denotado como $s_1 \prec s_2$) si y solo si s_1 es menor o igual en todas las funciones objetivo y existe al menos una función objetivo en la que s_1 es mejor que s_2. Es decir:

$$s_1 \prec s_2 \iff \forall k \in \{1, \ldots, n\}, f_k(s_1) \leq f_k(s_2) \wedge \exists k \in \{1, \ldots, n\}, f_k(s_1) < f_k(s_2) \tag{2.1}$$

La solución s_1 es la **solución dominante** y la solución s_2 es la **solución dominada**.

Cabe notar que si el par (s_1, s_2) no cumple con la condición de la Ecuación 2.1 se consideran como soluciones mutuamente no dominadas y se consideran incomparables. Es decir, $s_1 \nprec s_2 \wedge s_2 \nprec s_1$.

La meta en un problema multi-objetivo no es encontrar una sola solución, sino encontrar un conjunto de soluciones que sean mutuamente no dominadas y no dominadas por ninguna otra solución [23]. Cada nuevo conjunto de soluciones deberá mantenerse y actualizarse durante la búsqueda lo que es un proceso costoso debido al mayor número de comparaciones necesarias para ello.

Una solución no dominada por ninguna otra posible solución se considera una solución **Pareto óptima** [25]. Un conjunto de soluciones mutuamente no dominadas se considera como un **conjunto de Pareto**. Finalmente, el conjunto de soluciones Pareto óptimas se identifica como el **conjunto óptimo de Pareto** y la meta en un problema de optimización combinatoria multi-objetivo. El **Frente de Pareto** son los puntos en el espacio con la evaluación de las funciones objetivo de las soluciones de un conjunto de Pareto [23].

Es posible medir la calidad de un frente de Pareto conseguido por medio del **hipervolumen**, **hiperarea** o *S-metric* [23] [26]. Propuesto por [27], el hipervolumen es el área (o volumen)

cubierto por el frente de Pareto respecto del espacio. Mientras mayor sea, mejor es el conjunto de soluciones conseguido. Por tanto, maximizar el valor del hipervolumen es requerido.

El valor del hipervolumen de un frente de Pareto M es definido como [23]:

$$H(M) = \{ \bigcup_{m_i \in M} area_{m_i} \mid m \prec m_{ref} \} \tag{2.2}$$

Donde m_{ref} es una solución artificial arbitraria dominada por todas las posibles soluciones candidatas. $area_{m_i}$ es el área cubierta por el rectángulo que va desde cada solución m_i hasta m_{ref}. Como es descrito por [23], la Ecuación 2.2 define el hipervolumen como el área de la unión de los rectángulos formados.

En la Figura 2.2 se puede ver un ejemplo gráfico del hipervolumen de un problema donde se busca minimizar dos funciones $f_1(x)$ y $f_2(x)$. El frente de Pareto se compone por los puntos A, B, C, y D que corresponden a soluciones no dominadas entre sí. El valor del hipervolumen corresponde a la medida del área gris que cubren los rectángulos desde cada uno de dichos puntos hasta el punto de referencia x_{ref}. En este caso el valor del hipervolumen es 58.5

2.3. Problemas relacionados

El UTRP consiste en el diseño de una red de recorridos de buses de transporte público. Existen otros problemas que conllevan el diseño de una red no solo en otros contextos de transporte, sino que también en telecomunicaciones y sistemas de tuberías.

Adicionalmente, el diseño de un sistema de transporte público tiene muchos otros aspectos a considerar y se pueden identificar múltiples fases [6, 5, 28].

A continuación, se presenta una revisión de los distintos problemas relacionados con el UTRP.

Figura 2.2: Ejemplo gráfico de hipervolumen. El hipervolumen corresponde a la medida del área gris.

2.3.1. Problemas de diseño de red

En su forma más general, el UTRP es un caso particular de un problema de diseño de red aplicado al contexto de transporte público urbano. En este tipo de problemas se cuenta con nodos y posibles conexiones entre éstos, y se debe seleccionar cuales de estas posibles conexiones activar o incorporar en la red. Las conexiones seleccionadas son utilizadas para satisfacer las demandas de los usuarios asegurando flujos efectivos en costo. Problemas de este tipo aparecen en el diseño de redes de transporte y telecomunicaciones con diferentes formulaciones [29].

El UTRP se puede considerar un problema de este tipo puesto que, considerando los paraderos como nodos y los caminos como conexiones entre éstos, al crear las diferentes rutas se eligen los paraderos que componen esta ruta. Pero los paraderos de una ruta deben estar conectados, por tanto, se están eligiendo conexiones que activar para generar dicha ruta. Las rutas permiten a los usuarios viajar desde un punto a otro de la red.

En este tipo de problemas se pueden encontrar otros problemas de enrutamiento clásicos entre los que se incluye el Problema de Rutas de Vehículos (VRP, del inglés, *Vehicule Routing Problem*) que consiste en diseñar rutas de entregas con costo mínimo desde un depósito a clientes geográficamente dispersos sujeto a ciertas restricciones [30].

Otros problemas clásicos de optimización de redes son aquellos problemas de asignación de ubicaciones, por ejemplo, *p-median* y *p-center*, el Problema de Ubicación de Instalaciones No Capacitado, el Árbol Recubridor Mínimo, el Árbol de Steiner, Árbol de Comunicación Óptimo, y el Problema de Ubicación y Enrutamiento [29].

2.3.2. Problemas de transporte

Existen muchas versiones del problema concentradas en distintos aspectos en el diseño de un sistema de transporte y con distintos enfoques de resolución [5, 31, 28].

En [6], se reconocen 5 fases en el diseño de un sistema de buses de transporte: diseño de la red de buses, ajuste de frecuencias de los buses, desarrollo de horarios, itinerario de buses e itinerario de conductores. Cada fase se puede considerar un problema complejo por sí mismo, por lo que usualmente se resuelven independientemente en orden y el resultado de una fase es usado como entrada para la siguiente [6, 28].

En [31], se consideran aspectos de la red de tránsito; los caminos de la ciudad, dentro del sistema de transporte. Incluye aspectos como: construir y expandir calles, determinar orientación de calles de un sentido, asignar vías exclusivas para buses, asignación de carriles en las calles, itinerario del tránsito, luces de tránsito y de reparaciones de las calles.

En [28] se incorpora la idea de que fases posteriores deben retroalimentar con información

a las fases anteriores para el refinamiento de la solución. También, se separa el itinerario de conductores en la asignación de labores y la asignación de conductores a éstas. Además, se agrega una fase final de control en tiempo real del sistema de transporte para asegurar la eficiente operación de este frente a la incertidumbre en tiempos de viaje, patrones de demanda y problemas comunes: manifestaciones o marchas públicas, averías en vehículos, condiciones climatológicas y otras actividades más específicas.

Tanto en [31] como [28] se expone que para resolver un problema de transporte que involucre el diseño de la red o el ajuste de frecuencias, es necesario considerar la dependencia entre las decisiones de diseño y las decisiones de los usuarios. Las decisiones de los usuarios sobre el sistema de transporte, e.g. que recorridos tomar, que modos de transporte usar, etc. se van a ver influenciadas por el diseño de este, el nivel de servicio, la congestión del sistema, la información disponible y los sesgos personales que tenga cada usuario. Por tanto, el predecir que van a hacer los usuarios se vuelve un problema de asignación de tránsito. Es necesario resolver este problema para la evaluación del diseño de la red y las frecuencias.

Otro problema relacionado es el problema de asignación de tránsito que determina el patrón de flujo de la red de transporte para un escenario de diseño de red [31]. Esta asignación usualmente asume algún tipo de principios en el comportamiento de los usuarios para describir la elección de las rutas.

El criterio mayormente utilizado es el **principio de Wardrop** [31], estableciendo que los usuarios seleccionarán la ruta más corta. Así, se asume que todos los usuarios son no-cooperativos y conocen el tiempo de viaje actual para cada ruta. El patrón de flujo resultante es llamado el patrón de **Usuario-Equilibrio** (UE) en el que los pasajeros no tienen incentivo para cambiar a otras rutas, ya que al hacerlo sus tiempos de viaje aumentan. Esta asignación se conoce como asignación UE. De forma opuesta a UE, si se asume que los usuarios se comportan cooperativamente, la asignación resultante es llamada asignación optimizada para el sistema o **Sistema-Óptima** (SO).

La congestión de los vehículos del sistema es un aspecto incorporable en el problema de asignación de tránsito al considerar las capacidades de los vehículos. Esta congestión debería verse reflejada como un cambio en el comportamiento de los usuarios y sus elecciones de

ruta para los que se deben hacer suposiciones, lo que lleva a diferentes acercamientos para la asignación de pasajeros [31].

De forma general, tres enfoques se exponen en [28] que permiten manejar el problema de asignación de tránsito en acercamientos discretos:

- Descomponer el problema en sub-problemas secuenciales. Por ejemplo, definir rutas potenciales primero y luego resolver el problema de asignación de tránsito para la red propuesta e iterar esperando convergencia.
- Modelar el problema de transporte como un problema de optimización bi-nivel donde el diseño de las rutas es llevado a cabo en un "primer nivel" y la asignación de tránsito usada para determinar el flujo de pasajeros es determinada en un "nivel inferior".
- Modelar las decisiones de los pasajeros usando restricciones y funciones objetivo no lineales, llevando a problemas no-convexos que pueden ser resueltos a través de algoritmos heurísticos.

A continuación, se presenta una lista de los principales problemas de decisión dentro de un problema de transporte, categorizados en base al nivel de planificación:

- **Nivel Estratégico**: decisiones a largo plazo relacionadas a la infraestructura de las redes de transporte
  - El **Problema de Diseño de Red de Caminos** (**RNDP**, del inglés *Road Network Design Problem*) [31] consiste en determinar posibles proyectos de mejoras sobre una red de caminos, como construir nuevas calles o expandir las existentes de forma de minimizar los tiempos de viaje o reducir la congestión vehicular. En el marco de este problema existen decisiones de nivel táctico u operacional que pueden ser tomadas como asignar vías exclusivas para el transporte público o fijar una calle a un solo sentido.
  - El **Problema de Diseño de Red de Tránsito** (**TNDP**, del inglés *Transit Network Design Problem*) [31, 5, 28] consiste en determinar las rutas o recorridos

que conformaran la red de transporte público urbano sobre una red de caminos minimizando los tiempos de viajes de los usuarios, el coste de la red, etc.

- **Nivel Táctico**: utilización efectiva de la infraestructura y los recursos de las redes de transporte existentes definiendo niveles de servicio

  - El **Problema de Ajuste de Frecuencias** (**FSP**, del inglés *Frequency Setting Problem*) o **Problema de Itinerario de Red de Tránsito** (**TNSP**, del inglés *Transit Network Scheduling Problem*) [31, 28] consiste en ajustar las frecuencias de buses de los recorridos de una determinada red de transporte minimizando los tiempos de espera de los usuarios y el costo al operador.

  - El **Problema de Horario de Red de Tránsito** (**TNTP**, del inglés *Transit Network Timetabling Problem*) [31, 28] consiste en el desarrollo de los horarios de los recorridos, es decir, determinar las horas de salida y llegada de los buses a los distintos paraderos en la red de forma de minimizar el tiempo de espera de los usuarios. Cabe notar que este aspecto no es contemplado en todos los sistemas de transporte.

- **Nivel Operacional**: decisiones a corto plazo de asignación de recursos e itinerario para cumplir con los niveles de servicios requeridos por el sistema

  - El **Problema de Itinerario de Vehículos** (**VSP**, del inglés *Vehicle Scheduling Problem*) [28] consiste en asignar los diferentes vehículos disponibles a cada uno de los viajes que se realizaran. Existen distintos tipos de vehículos con diferentes capacidades y consumo de combustible. Se debe minimizar el consumo de combustible y los tiempos de espera de los pasajeros.

  - El **Problema de Itinerario de Conductores** (**DSP**, del inglés *Driver Scheduling Problem*) consiste en asignar las labores diarias a cubrir de un bloque de viajes de vehículos para minimizar el costo de estas labores y cumplir con restricciones de leyes laborales.

  - El **Problema de Nómina de Conductores** (**DRP**, del inglés *Driver Rostering Problem*) determina la asignación de conductores para las distintas labores determinadas por el DSP para un período de planificación cumpliendo con leyes

laborales y normas de la compañía.

Cabe notar que algunos de estos problemas se resuelven conjuntamente. Por ejemplo, es usual en la literatura ver mencionado el Problema de Diseño de Red e Itinerario de Tránsito (**TNDSP**, del inglés *Transit Network Design and Scheduling Problem*) [28, 5, 31] que consiste en conjuntamente llevar a cabo el diseño de una red de recorridos de buses y el ajuste de frecuencias de éstos, es decir, una combinación del TNDP y TNSP [5].

2.4. Hacia el UTRP

Como se vio anteriormente, el TNDP consiste en determinar las rutas para una red de transporte público sobre una red de caminos optimizando algún criterio sujeto a ciertas restricciones. Cuenta con una gran variedad de enfoques, donde se usa tanto aproximación con variables continuas como optimización de variables discretas [28].

Entre los criterios a optimizar en la literatura se encuentran: minimizar tiempo de viaje promedio de los usuarios, minimizar costes de operación, minimizar tamaño de la flota, minimizar costes de construcción, minimizar tiempos de espera, maximizar número de viajes directos, minimizar número de transbordos y maximizar demanda satisfecha [28, 5, 31].

Las restricciones son igual de variadas. Algunos de los objetivos antes vistos, se consideran restricciones en otros acercamientos. Entre las que han incluido en trabajos anteriores se encuentran: satisfacción completa de la demanda, número, largo y factibilidad de rutas, tamaño de la flota, capacidad de los vehículos, presupuesto, frecuencia mínima, conjunto restringido de rutas, cobertura de área y forma de la red [28, 5, 31].

El UTRP es un caso particular de TNDP. La naturaleza de todas las variables es discreta. Contempla demanda y tiempos de viaje deterministas, con los usuarios viajando a su destino a través de la ruta más corta sin considerar los niveles de congestión o capacidad vehicular. El UTRP desde trabajos recientes se considera un problema multi-objetivo con dos criterios a optimizar: minimizar el tiempo de viaje promedio de los pasajeros y el costo total de la red [8, 12, 14]. Las restricciones son sencillas: hay un número predefinido de rutas, las cuales

tienen un límite mínimo y máximo en su largo en número de nodos; las rutas deben satisfacer toda la demanda conectando todos los nodos de la red de caminos y no contener ciclos ni retrocesos.

A continuación, se presentan acercamientos y técnicas que resuelven el UTRP o versiones muy similares.

Uno de los primeros trabajos sobre el UTRP encontrados data del año 1979 [32] siendo uno de los trabajos pioneros para el problema. Se resolvió el problema en su forma general de manera iterativa, primero produciendo el set de rutas para después asignar los vehículos y los pasajeros a estos. Se usaba una técnica iterativa a partir de un set de rutas factibles. El algoritmo fue aplicado a una ciudad de Bélgica con 150.000 habitantes. La red constaba de aproximadamente 130 nodos, 350 enlaces y 12 rutas. El algoritmo tenía un tiempo de ejecución considerable (60 minutos).

En 2010 [10], se diseñó un *framework* para la resolución de un UTRP mono-objetivo planteando una representación sencilla y un operador básico *Make-Small-Change* para agregar y eliminar nodos en los extremos de las rutas. Estos fueron probados usando *Hill-Climbing* y *Simulated Annealing* buscando optimizar el tiempo de viaje y número de transbordos. Consiguieron mejores resultados que trabajos anteriores. Notaron que el operador usado era menos efectivo mientras se restringiera más el largo de las rutas.

En 2013, [8] extendió el trabajo [10] para la implementación de un algoritmo evolutivo que tomaba en cuenta dos objetivos: el tiempo de viaje promedio y el costo al operador. Cruzaban las soluciones mezclando las rutas de dos conjuntos distintos. Para la mutación hacían uso del operador *Make-Small-Change*. Nuevas soluciones eran incorporadas a la población de soluciones si dominaban a las soluciones progenitoras, sino eran descartadas. Consiguieron mejoras respecto del trabajo anterior y crearon frentes de Pareto para varias instancias, algunas de ellas nuevas.

En 2014 [2], se implementó una optimización por enjambre de partículas para el UTRP optimizando el tiempo de viaje y número de transbordos. Estudiaron el efecto de incluir en el proceso de optimización el costo al operador asociado a la longitud de las rutas. Propusieron dos movimientos para las partículas: intercambiar partes de una ruta que tengan un nodo en

común evitando ciclos e intercambiar dos rutas completas al azar entre los conjuntos. Consiguieron mejores resultados, aunque muy cercanos en cuanto a tiempo de viaje y porcentaje de usuarios satisfechos sin transbordos. Notaron que el limitar el costo al operador afectaba negativamente el tiempo de viaje. También vieron que al variar la importancia del número de transbordos la calidad de las soluciones no se veía afectada significativamente.

En 2014 [12], extendió el algoritmo genético de [8], presentando una nueva heurística de construcción y nuevos operadores de mutación: intercambiar partes, mezclar, reemplazar, invertir partes y remover traslapes entre las rutas de un conjunto. Todos estos cambios fueron aplicados al algoritmo anterior y en un framework NSGAII para apoyar la obtención de mejores frentes de Pareto. Consiguieron mejoras al aplicar las nuevas heurísticas en el algoritmo anterior, pero los mejores resultados fueron conseguidos al hacer uso de NSGAII.

En 2014 [13], se encargó de la paralelización del algoritmo anterior con la meta de mejorar los tiempos de ejecución. Llevó a cabo dos tipos de paralelización: una a nivel de cruzamientos y evaluaciones, entregando cada cruzamiento y evaluación como un *job* separado, la otra fue a nivel de las poblaciones separando ésta en varias islas donde cada isla podía correr independiente de las otras. Logró mejoras considerables en el tiempo de ejecución que mejoraban aún más a medida que aumentaba el número de núcleos asignados a la ejecución del algoritmo.

En 2019 [14], se resolvió el UTRP con hiper-heurísticas de selección. Contaban con una fase de selección de heurísticas a aplicar y una fase de aceptación en que se actualizaban los parámetros que controlaban la selección dependiendo de la calidad de la solución generada. Para evaluar las soluciones ponderaban las restricciones y los objetivos. Observaron que la combinación de métodos de selección y aceptación de movimientos que daba mejores resultados era Selección basada en Secuencia con *Great Deluge*. Lograron obtener mejores soluciones en casi todas las instancias.

En 2019 [16] resolvió el UTRP incorporando una restricción de nodos terminales: los nodos en cada extremo de las rutas debían corresponder a un nodo terminal. La optimización era realizada usando un algoritmo genético NSGAII adaptado para el manejo de los nodos terminales. Para la generación de la población inicial se generaba un mapa de uso de los enlaces

por ruta más corta: se calcula la ruta más corta entre todos los pares de nodos para luego sumar la cantidad de usuarios que van a través de cada enlace. Usando esta información generaban conjuntos de rutas candidatas. La implementación del algoritmo NSGAII era similar al de [12] aunque con operadores de mutación especializados. Adicionalmente se intentaba aplicar un algoritmo de reparación si una solución generada era infactible debido a nodos desconectados o rutas solapadas. Compararon el rendimiento de sus rutas con aquellas del caso real, obteniendo mejores resultados.

En términos de encontrar soluciones que optimicen el tiempo de viaje de los pasajeros la hiper-heurística de [14] entrega los mejores resultados disponibles en la literatura. Sin embargo, para encontrar múltiples soluciones de un frente de Pareto requiere múltiples ejecuciones con la variación de un parámetro del algoritmo.

En cuanto a frentes de Pareto, el algoritmo evolutivo de [13] es aquel que ha publicado los mejores resultados en esta revisión necesitando una sola ejecución para generar el frente de Pareto.

2.5. Formalización de UTRP

A continuación, presentamos la formalización del problema pertinente para el presente trabajo.

El UTRP considera un grafo no dirigido de n nodos, donde cada nodo representa alguna ubicación de interés o un punto de acceso (por ejemplo, paraderos o estaciones), y los enlaces las conexiones directas entre estos. Una instancia del problema requiere incluir tres elementos básicos: la Red de Caminos, la Matriz de Demanda y los Parámetros de las Rutas.

2.5.1. Elementos básicos

Primero, la **Red de Caminos** corresponde a la estructura de los caminos de la ciudad y las ubicaciones que deberán ser cubiertas por los buses. Se representa como un grafo no dirigido

donde el peso de los enlaces corresponde al tiempo de viaje en minutos, es decir, cuánto demoran los pasajeros en viajar entre los nodos. El tiempo de viaje se considera simétrico, es decir, se considera como si un usuario se demorará lo mismo viajando en ambas direcciones.

Definición 2.1 (Red de Caminos). Dado un conjunto de nodos $N = \{n_1, \cdots, n_m\}$ y un conjunto de enlaces $L = \{(n_i, n_j) \mid n_i, n_j \in N, conectados(n_i, n_j) \wedge n_i \neq n_j\}$. Entonces, una **Red de Caminos** R es un grafo no dirigido $R = (N, L)$ tal que cada nodo n_i es una ubicación y cada enlace $(n_i, n_j) \in L$ es un camino que conecta directamente las ubicaciones n_i y n_j.

Es posible representar este grafo usando una matriz en donde cada entrada represente el tiempo de viaje para viajar entre nodos conectados.

Definición 2.2 (Matriz de tiempos de viaje). Sea $R = (N, L)$ una red de caminos con un conjunto de nodos $N = \{n_1, \ldots, n_m\}$. La **matriz de tiempos de viaje** C es una matriz simétrica de dimensión $|N| \times |N|$. Cada entrada $C_{ab} \in \mathbb{Z}$ representa el tiempo de viaje, entre los nodos n_a y n_b. $C_{ab} \geq 0$ cuando exista conexión entre los nodos, mientras que $C_{ab} = \infty$ cuando no.

La **Matriz de Demanda** representa la demanda de transporte, esto es, cuántos pasajeros desean trasladarse desde una ubicación a otra. En general, se considera de forma atemporal (que no depende de ciertas horas del día, temporadas del año o eventos). Asimismo, el valor es simétrico puesto que es muy probable que un usuario que viaje hacia un nodo quiera regresar más tarde al punto de partida.

Definición 2.3 (Matriz de demanda). Dada una Red de Caminos $R = (N, L)$ con un conjunto de nodos $N = \{n_1, \ldots, n_m\}$. Se define $D_{|N| \times |N|}$ como la **Matriz de Demanda**. Cada entrada $D_{ab} \in \mathbb{Z}$ indica el número de pasajeros que viajan desde el nodo n_a hasta el nodo n_b. Notar que $D_{aa} = 0 \ \forall n_a \in N$.

Los **Parámetros de las Rutas** establecen restricciones típicas de cada ruta y son definidos por el planificador de la red de transporte de acuerdo a los límites de costo al operador. Estos

parámetros contemplan el largo de las rutas y el número de estas.

Definición 2.4 (Parámetros de las rutas). Se define q como la cantidad de rutas que debe tener la solución. También se definen $\lambda_{min}, \lambda_{max} \in \mathbb{Z}$ como el número mínimo y máximo de nodos que debe tener cada ruta, respectivamente.

2.5.2. Evaluación de Soluciones

Una solución para el UTRP requiere que se construya una **Red de Rutas** de transporte. Cada ruta de la red corresponde a un vector ordenado de nodos que están conectados en la Red de Caminos.

Definición 2.5 (Red de Rutas). Dada una Red de Caminos $R = (N, L)$ con un conjunto de nodos $N = \{n_1, \ldots, n_m\}$ y conjunto de enlaces L.

Se define una **Ruta** $W = [\mu_1, \mu_2, \ldots, \mu_v]^T$ como un vector ordenado de v nodos tal que $\mu_i \in N$ $\forall i \in \{1, \ldots, v\}$ y $(\mu_i, \mu_{i+1}) \in L$ $\forall i \in \{1, \ldots, v-1\}$.

Entonces, se define una **Red de Rutas** U como un conjunto de q rutas.

$$U = \{W_1, W_2, \ldots, W_q\} \tag{2.3}$$

Las rutas deben construirse cumpliendo un conjunto de objetivos entre los que pueden incluirse: satisfacer la mayor cantidad de demanda de transporte de los pasajeros que sea posible, minimizar los tiempos de viaje, minimizar el número de transbordos entre rutas, minimizar el costo de operación, minimizar el largo de las rutas, maximizar los ingresos o las ganancias, etc.

En esta formulación se considera la optimización del **Tiempo de viaje de los pasajeros** y **costo al operador**.

El **Tiempo de viaje de los pasajeros** es lo que demoran en viajar todos los pasajeros entre cada par de nodos. Se asume que el recorrido de los pasajeros entre dos nodos consiste en el camino más corto entre éstos haciendo uso de la red de transporte. Se considera el tiempo de viaje promedio para comparar más fácilmente con números de menor magnitud [10, 8].

Cabe notar que se debe considerar el número de transbordos o transferencias para el cálculo de ruta más corta. Obviamente cada transbordo tiene un impacto en el tiempo total de viaje de un pasajero debido al tiempo de desplazamiento entre paraderos y el de espera del siguiente bus. Este tiempo es considerado con un valor fijo de 5 minutos en múltiples trabajos anteriores [8, 2, 14].

Definición 2.6 (Función Objetivo de Tiempo de Viaje Promedio). Dada una Red de Caminos $R = (N, L)$ con un conjunto de nodos $N = \{n_1, \dots n_m\}$ asociada a una Matriz de Tiempos de Viaje C. Para R se ha construido una Red de Rutas $U = \{W_1, \dots, W_q\}$.

Se define un **Recorrido** T_{ab} como un vector de nodos ordenados que, a través de los nodos conectados por la Red de Rutas U, señala una secuencia de f nodos que permite viajar desde el nodo n_a hasta el nodo n_b.

$$T_{ab} = [t_1, t_2, \dots, t_f]^T, t_i \in N \ \forall i \in \{1, \dots, f\}, t_1 = n_a, t_f = n_b \tag{2.4}$$

Para cada par de nodos consecutivos en el recorrido T_{ab} existe una Ruta W que conecta dichos nodos.

$$\forall i \in \{1, \dots, f-1\},\ \exists W = [\mu_1, \dots, \mu_v]^T \in U,\ \exists j \in \{1, \dots, v-1\}, t_i = \mu_j, t_{i+1} = \mu_{j+1} \tag{2.5}$$

Para cualquier recorrido $T_{ab} = \{t_1, \dots, t_f\}$, se tiene que para algún $i \in \{1, \dots, f-1\}$, y siendo $n_x = t_i$ y $n_y = t_{i+1}$, entonces se define $\hat{C}_i = C_{xy}$ para obtener el tiempo de viaje entre cualquier par de nodos consecutivos t_i y t_{i+1} del Recorrido T_{ab}. Con todo esto, se denota el **Costo del Recorrido** T_{ab} como $K(T_{ab})$ y se define como la suma de los tiempos de viaje entre cada par de nodos consecutivos de dicho recorrido.

$$K(T_{ab}) = \sum_{i=1}^{s-1} \hat{C}_i \tag{2.6}$$

También se define $H(T_{ab})$ como el número de transbordos que son requeridos en el recorrido T_{ab}, esto es, la cantidad de veces que fue necesario cambiar a otra ruta para continuar con el recorrido.

Siendo α la ponderación para el tiempo de viaje y β la ponderación del número de transbordos en el costo de los pasajeros. Se considera a $\hat{T}_{ab}$ como el **Recorrido Mínimo** entre los nodos n_a y n_b que tiene menor tiempo de viaje:

$$\nexists T_{ab}, \alpha K(T_{ab}) + \beta H(T_{ab}) < \alpha K(\hat{T}_{ab}) + \beta H(\hat{T}_{ab}) \tag{2.7}$$

Por último, sea D la Matriz de Demanda, se denota como C_p el **Tiempo de viaje promedio** de los pasajeros y se define como el tiempo de viaje promedio de todos los pasajeros asumiendo que viajaran a través de sus respectivos Recorridos Mínimos desde su origen a su destino. En el UTRP se busca la Red de Rutas U que minimice este valor.

$$\text{mín} : C_p = \frac{\alpha \sum_{i,j=1}^{m} D_{ij} K(\hat{T}_{ij}) + \beta \sum_{i,j=1}^{m} D_{ij} H(\hat{T}_{ij})}{\sum_{i,j=1}^{m} D_{ij}} \tag{2.8}$$

El **Costo del operador** es una medida del costo de implementar una Red de Rutas. Se considerará como la suma de los tiempos de viaje de las rutas puesto que el costo del operador aumentará proporcionalmente junto a este valor. Mientras más larga sea cada ruta, se asume que más costoso será el sistema para el operador debido a la mayor cantidad de buses que requiere un conjunto de rutas más largas para mantener una frecuencia y nivel de servicio aceptables.

Definición 2.7 (Función Objetivo de Costo al operador). Dada una Red de Rutas $U = \{W_1, \ldots, W_w\}$ y una Matriz de tiempos de viaje C. Notar que $W = [\mu_1, \ldots, \mu_v]$ $\forall W \in U$. Para algún $i \in \{1, \ldots, v-1\}$ y siendo $n_a = \mu_i$ y $n_b = \mu_{i+1}$, se define $\hat{C}_i = C_{ab}$ para obtener el tiempo

de viaje entre nodos consecutivos μ_i y μ_{i+1} de la ruta W.

Se denota como C_o, el **Costo del operador** y es definido como la suma de los tiempos de viaje de los enlaces que forman las rutas. En el UTRP se busca la Red de Rutas U que minimice este valor.

$$\text{mín} : C_o = \sum_{W \in U} \sum_{i=1}^{|W|-1} \hat{C}_i \tag{2.9}$$

Para que una Red de Rutas se considere como una solución factible debe cumplir las siguientes restricciones:

- Sus rutas deben tener un largo dentro de un intervalo predeterminado
- Debe asegurarse la conectividad de todos los nodos
- Cada ruta no debe tener ningún ciclo ni retroceso.

Estas restricciones quedan formalizadas en la Definición 2.8.

Definición 2.8 (Restricciones UTRP). Sea $R = (N, L)$ una Red de Caminos para la que se ha construido una Red de Rutas $U = \{W_1, W_2, \ldots, W_w\}$.

Cada Ruta $W \in U$ estará restringida a tener un mínimo λ_{min} y máximo λ_{max} de nodos.

$$\lambda_{min} \leq |W| \leq \lambda_{max} \; \forall W \in U \tag{2.10}$$

La Red de Rutas U debe ofrecer conectividad completa de la Red de Caminos permitiendo viajar entre cualquier par de nodos, es decir, para cualquier par de nodos n_a y n_b debe haber un Recorrido T_{ab} válido.

$$\exists T_{ab} \forall n_a, n_b \in N \tag{2.11}$$

Por último, se asumirá que no habrán ni ciclos ni retrocesos en ninguna de las rutas $W = [\mu_1, \ldots, \mu_v]^T \in U$ para lo que simplemente se requiere que no se repita ningún nodo.

$$\mu_i \neq \mu_j \forall i, j \in \{1, \ldots v\} \; \forall W \in U \tag{2.12}$$

Se puede ver que la formulación presentada para el UTRP es relativamente sencilla y no toma en cuenta una multitud de factores como: tiempos de espera y abordaje, congestión de la red, capacidad de los buses, nodos terminales, coste no lineal de combustible, etc.

Sin embargo, no por eso el problema debería considerarse trivial pues aún se clasifica como un problema de diseño de red quedando dentro de la clase de complejidad NP-Hard [7]. Es por esto que en la literatura estudiada en la sección anterior se advierte la presencia total de metaheurísticas que solo son capaces de aproximar la solución.

La evaluación de soluciones para el UTRP ha demostrado abarcar una parte significativa del tiempo de ejecución de cualquier algoritmo de resolución [15]. Esto es debido al componente $\hat{T}_{ij}$ de Recorrido Mínimo usando la Red de Rutas que puede ser encontrada en la Definición 2.6 en la fórmula de Tiempo de Viaje Promedio. Esta componente requiere el cálculo del Recorrido Mínimo entre todos los pares de nodos usando la Red de Rutas y tomando en cuenta los transbordos. Por tanto, será requerido realizar este cálculo para la evaluación de cualquier solución generada, teniendo un alto tiempo de ejecución en comparación a la otra función objetivo y al chequeo de restricciones.

2.6. Análisis de Instancias previas del UTRP

Como se indica en [15, 8] la mayoría de las instancias de UTRP son ficticias, generadas aleatoriamente o simplificadas. Esto lleva a una aproximación imprecisa de los casos de la vida real. En la Tabla 2.2 se presentan instancias UTRP conocidas y sus características. Hay que notar que cada característica influye en la dificultad de resolución y puede ser vista como una **Métrica de Complejidad** de la instancia.

LB_{ATT} es la cota inferior del tiempo promedio de viaje como se calcula en [8, 33] y MST el coste del árbol de cobertura mínima (y una cota inferior al costo del operador).

A continuación, mostramos un análisis enfocado en la metodología aplicada para generar cada una de estas instancias.

Tabla 2.2: Resumen de instancias UTRP.

Instancia	**Nodos**	**Enlaces**	**Rutas**	$\lambda_{min} - \lambda_{max}$	LB_{ATT}	MST
Mandl [34]	15	21	4-8	2-8	10.0058	63
Mumford0 [8]	30	90	12	2-15	13.0121	94
Mumford1 [8]	70	210	15	10-30	19.2695	228
Mumford2 [8]	110	385	56	10-22	22.1689	354
Mumford3 [8]	127	425	60	12-25	24.7453	394
Nottingham100 [15]	100	187	40	10-25	19.2974	254
Edinburgh200 [15]	200	362	90	5-25	20.8211	476
NottinghamR [16]	376	656	69	3-52	7.1003	418

Las métricas de complejidad más directas son la cantidad de nodos y enlaces: mientras mayor sean, el espacio de búsqueda de la instancia y el tiempo de evaluación de la solución van a incrementar considerablemente. Así la instancia Mandl, la más usada para la comparación de algoritmos, es la más fácil; en contraparte, NottinghamR es la más compleja. Sin embargo, todas estas instancias son muy pequeñas respecto a grandes ciudades que tienen miles de paraderos o ubicaciones, por ejemplo, el sistema RED de Santiago de Chile que incorpora más de 10000 nodos.

La cantidad y largo de las rutas, comúnmente definidas por el diseñador de la red, restringen críticamente el espacio de soluciones factibles. Mientras mayor sea la cantidad y el largo de las rutas, lleva a un espacio de soluciones más grande; en contraparte, mientras menor sean estos valores, más restringido será el espacio de búsqueda. Así, la diversificación es directamente afectada, llevando a una convergencia lenta o prematura.

Las métricas LB_{ATT} y MST son respectivamente las cotas inferiores del tiempo de viaje de

los pasajeros y los costos al operador del sistema. Estas métricas muestran la complejidad de cada instancia en ambos criterios de resolución, esto es, valores pequeños en ambas métricas implican un mayor esfuerzo requerido para la resolución u optimización.

La forma en que cada instancia fue generada es relevante, puesto que los valores de las métricas pueden cambiar significativamente. Por ejemplo, los valores de LB_{ATT} y MST varían considerablemente entre [15] y [16] a pesar de que cubren el mismo caso real y esto se debe a una diferencia en el método de generación. El primer acercamiento usa un pequeño subconjunto tomado desde los datos originales, lo que conlleva a valores más pequeños para LB_{ATT} y más grande para MST debido a las mayores distancias entre las ubicaciones y a la menor cantidad de nodos, respectivamente. El segundo acercamiento incluye información de cada esquina, lo que incrementa la cantidad de distancias corta de viaje y, consecuentemente, la cantidad de nodos. Por lo que, se asignan valores menores para LB_{ATT} y mayores para MST.

La tabla 2.3 resume los métodos seleccionados para generar cada instancia UTRP discutida en este capítulo.

Mumford [8] creó cuatro instancias disponibles públicamente. Fueron generadas basadas en parámetros predefinidos para la red de caminos (número de nodos, enlaces y enlaces por nodo) y la matriz de demanda (límite inferior y superior). Las coordenadas de los nodos se definieron dentro de un área rectangular y sus valores generados a partir de una distribución aleatoria uniforme. El tiempo de viaje estaba basado en distancia Euclidiana y la demanda entre los nodos era aleatoria entre valores predefinidos. Los enlaces fueron escogidos apoyándose en un árbol de cobertura mínima para asegurar conectividad de la red, pero luego nodos extra eran escogidos al azar y agregados a la red, a través de los enlaces más cortos, para alcanzar los valores deseados.

Como estas instancias están basadas principalmente en procedimientos aleatorios para la colocación de los nodos y generación de la matriz de demanda, difícilmente se ajustan a escenarios reales y son bien ficticias.

John et al [15] produjo dos instancias de manera similar. Los paraderos eran escogidos al azar con sus coordenadas extraídas desde el *United Kingdom National Public Transport Data Repository*. Los tiempos de viaje estaban basados en datos obtenidos desde la API de matriz

Tabla 2.3: Generación de elementos básicos de las instancias UTRP

Instance	**Colocación de nodos**	**Información de enlaces**	**Matriz de demanda**	**Comentario**
Mandl	Método desconocido. Información basada en red Suiza [34].			Instancia más pequeña.
Mumford0 Mumford1 Mumford2 Mumford3	Coordenadas aleatorias.	Distancia Euclidiana.	Valores aleatorio uniformes.	No se ajusta a escenarios reales.
Nottingham100 Edinburgh200	Paraderos escogidos al azar desde el *UK National Public Transport Data Repository*.	API de distancia de Google.	Datos censales.	La selección al azar de paraderos puede dejar áreas de baja densidad desconectadas.
NottinghamR	Mezcla de esquinas. Basado en la *UK Ordnance Survey* .	Función ”Instalación más cercana” de ArcGIS.	Datos censales.	Datos de la vida real, pero relativamente pequeños respecto a ciudades grandes. La reducción se limita a mezclar nodos que representan las mismas ubicaciones y calles.

de distancia de Google. La demanda era calculada relacionando el número de pasajeros a la densidad de la población en el área cuadrada de $1 \times 1Km^2$ que rodeaba cada paradero.

Aunque estas nuevas instancias incorporan datos reales, el procedimiento de selección aleatoria tiende a elegir nodos ubicados en áreas de alta densidad. Por tanto, nodos que se encuentren en áreas periféricas o de baja densidad tendrán tendencia a no ser escogidos quedando no representados en la red de caminos.

Soares et al [16] generó una instancia para el caso de Nottingham con una red de caminos adaptiva. Para ello los nodos eran colocados en esquinas, intersecciones, rotondas, etc. los que luego se mezclaban si se encontraban a una cierta distancia entre ellos, conformándose la red por nodos intermedios a estos nodos mezclados. Nodos adicionales eran agregados si es que entre un par de nodos intermedios un área de demanda quedaba sin ser cubierta por la red. Los enlaces y pesos de la red de caminos se determinaban con la función ”Instalación más cercana” del software ArcGIS que calcula la ruta más cercana entre ambos nodos.

Para generar la matriz de demanda se basaron en datos censales del Reino Unido que datan del 2011 que incluía información de origen destino desde áreas de origen hasta zonas con puestos de trabajo. La demanda de una zona de origen o destino era asignada a un nodo, si el centroide de la zona se encontraba dentro de un área circular rodeando al nodo. Si el centroide de la zona se encontraba dentro de múltiples de estas áreas, la demanda era repartida equitativamente entre los nodos.

La reducción usada permitía mantener la estructura de la red de caminos y estaba basada completamente en datos de la vida real, pero era aún muy pequeña respecto a ciudades grandes que incluyan miles de paraderos y/o ubicaciones.

La técnica de agrupación de nodos propuesta logró reducir la cantidad de nodos a 376 desde 428. Aunque esto implico una ganancia en tiempo de ejecución, es insuficiente para casos masivos con miles de nodos, pues fue incapaz de reducir el orden de magnitud de la cantidad de nodos.

Concluyendo, este análisis muestra que la mayoría de las instancias fueron generadas con métodos aleatorios o que dependen de ciertas condiciones en la distribución de los paraderos,

por lo que no se ajustan a casos reales.

La instancia más grande, NottingHamR tiene 376 nodos que es un número considerablemente menor que el número de nodos de una ciudad de tamaño considerable como es el caso del sistema RED en Santiago de Chile que cubre más de 10000 paraderos.

Por tanto, una técnica de relajación para escenarios complejos o intractables es requerida si se desea resolverlos usando los algoritmos de resolución del estado del arte. Esta técnica de relajación debe mantener una estructura de caminos similares que mantenga conectadas las zonas periféricas o de baja densidad y cuya distribución de demanda tenga un comportamiento similar al caso real.

Capítulo 3

Algoritmos de clustering y similitud

3.1. Algoritmos de clustering

Las técnicas de clustering agrupan un grupo de objetos (datos, puntos, documentos, observaciones, etc.) en subconjuntos o clústeres. La meta de los algoritmos es crear clústeres que sean coherentes internamente, pero que sean claramente diferentes de cada otro [35].

El framework de creación de instancias utiliza técnicas de clustering para detectar grupos de paraderos cercanos y generar uno que los represente.

En esta sección, primero se presentan un par de definiciones básicas, seguido de una revisión de las distintas técnicas de clustering, finalizando con la presentación de métricas para medir la similitud de la salida de algoritmos de clustering.

3.1.1. Clustering

En este trabajo los objetos o elementos que se desearan agrupar serán ubicaciones, que serán agrupadas en varios grupos o *clústeres* por medio de algún *algoritmo de clustering*.

Cada ubicación tendrá asociada una cierta posición en el espacio y cada clúster podrá ser

representado por el centroide como el promedio de las posiciones de las ubicaciones en este.

Definición 3.1 (Centroide). Dado un conjunto de ubicaciones S, donde cada elemento $s \in S$ tiene una posición $pos(s)$. El **centroide** del conjunto S es la posición promedio entre todos sus elementos.

$$C_S = \frac{1}{|S|} \sum_{s \in S} pos(s) \tag{3.1}$$

A partir de las posiciones de las ubicaciones, las técnicas de clustering agruparan las ubicaciones en uno o más grupos o clústeres, dejando en un mismo clúster aquellas ubicaciones que se encuentren cerca entre sí.

Cabe destacar que los clústeres son conjuntos de ubicaciones exclusivos los unos con los otros, esto es, que cada ubicación pertenecerá solamente a un clúster.

Definición 3.2 (Clústeres). Dado un conjunto de elementos ubicaciones S. Un algoritmo de clustering α genera un **conjunto de clústeres** A generando m subconjuntos de ubicaciones.

$$A = \left\{ a_i \;\middle|\; a_i \subset S \;\wedge\; a_i \cap a_j = \emptyset \;\forall i \neq j \;\wedge\; \bigcup_{1 \leq i \leq m} a_i = S \right\} \tag{3.2}$$

Así, cada **clúster** $a_i \in A$, esto es, un conjunto de ubicaciones, tiene su propio centroide C_{a_i}. Desde aquí en adelante, se reduce la notación a $a \in A$ para cada clúster y, por consiguiente, se entiende que $|A| = m$.

3.1.2. K-Means

El algoritmo *K-Means* agrupa los datos por medio de actualizaciones sucesivas de los centroides de los clústeres. Para ello comienza con K centroides en posiciones arbitrarias y asigna los puntos más cercanos a cada centroide como un clúster. Luego, a partir de esta

asignación, se actualiza la posición de los K centroides, volviendo a iterar.

El algoritmo agrupa los datos tratando de separarlos en grupos de igual varianza, minimizando el promedio de la distancia Euclidiana cuadrada al centro del clúster [36, 35].

Por tanto, dado un entero k y un conjunto de n puntos de datos $X \subset \mathbb{R}^d$ que se desean agrupar. Se quiere encontrar un conjunto C de k centros para minimizar la función potencial definida en Ecuación 3.3.

$$\phi = \sum_{x \in X} \min_{c \in C} \|x - c\|^2 \tag{3.3}$$

Cada punto quedará asignado a un centro con lo que los puntos quedarán agrupados.

El algoritmo *K-Means* opera según lo indicado en el Algoritmo 1 [36].

Algoritmo 1 Algoritmo de clustering K-means

1. Elegir arbitrariamente un conjunto $C = \{c_1, c_2, \ldots, c_k\}$ de k centroides. Es práctica estándar elegir los centroides iniciales de manera aleatoria uniforme desde X.

2. Por cada $i \in \{1, \ldots, k\}$, establecer el clúster C_i como el conjunto de puntos en X que estén más cercanos a c_i que lo que están a c_j con $j \neq i$.

3. Por cada $i \in \{1, \ldots, k\}$, establecer c_i como el centroide de todos los puntos en C_i: $c_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$.

4. Repetir los Pasos 2 y 3 hasta que C no cambie o algún otro criterio de término.

El algoritmo *K-Means* converge en el sentido de que la distancia de los puntos de cada clúster al centroide disminuye (o se mantiene) en cada iteración. No obstante, no hay garantía de que se alcance un mínimo global de la función potencial [35].

3.1.3. Clustering Jerárquico Aglomerativo

Existe la familia de algoritmos de clustering jerárquicos que se compone de algoritmos basados en la construcción de clústeres anidados, producto de la mezcla (o separación) sucesiva de clústeres dependiendo de la distancia entre elementos.

Esta anidación de clústeres genera una jerarquía que puede representarse usando un árbol o dendrograma. La raíz del árbol es el clúster único que agrupa todos los objetos, y las hojas son los clústeres con un único objeto. Esto permite obtener una jerarquía de clústeres más informativa, aunque a cambio de un mayor tiempo de ejecución en comparación a *K-Means* [35].

El Clustering Aglomerativo opera desde abajo hacia arriba: cada objeto empieza solo en su propio clúster, para luego sucesivamente mezclarse con otros clústeres [37] [35]. Cada mezcla es decidida con un criterio que elija aquellos clústeres más similares. Existen varios criterios de mezcla o vinculación (*linkage*) también conocidos como medidas de similitud:

- ***Ward***: minimización de la suma de las diferencias cuadradas dentro de todos los clústeres (la función potencial Ecuación 3.3) teniendo un enfoque similar a *K-means*.
- **Vinculación máxima** o **completa**: minimiza la máxima distancia entre las observaciones de pares de clústeres.
- **Vinculación promedio**: minimiza el promedio de las distancias entre todos los objetos de pares de clústeres.
- **Vinculación singular**: minimiza la distancia entre los objetos más cercanos de pares de clústeres.

Una versión simplificada del algoritmo se puede observar en el Algoritmo 2 [35].

Algoritmo 2 Algoritmo de clustering aglomerativo

1. Asignar a cada objeto su propio clúster
2. Computar una matriz de similitud $C_{N \times N}$ donde N es el número de objetos a agrupar.
3. Realizar $N - 1$ iteraciones (en el caso de que se mezclen todos hasta conseguir un solo clúster) o hasta que se alcance un criterio de parada. En cada iteración:
   a) Mezclar los clústeres i y m más similares según lo indicado por C.
   b) Actualizar $C[i][j]$ y $C[j][i]$ a $sim(i, m, j)$ que indica la similitud del clúster j con la mezcla de los clústeres i y m.
4. Retornar una lista con las mezclas realizadas.

3.1.4. Clustering espectral

Este tipo de clustering lleva a cabo una reducción de dimensionalidad sobre un grafo de similitud asociado a los datos. Para ello calcula los vectores propios de la matriz Laplaciana del grafo de similitud, sobre los cuales se aplica otra técnica de clustering como *k-means* para obtener los distintos clústeres [38].

La reducción de dimensionalidad genera una representación alternativa que resalta características latentes del conjunto de datos permitiendo agrupar en base a estas.

El grafo de similitud $G = (V, E)$ corresponde a una representación de grafo de los objetos a agrupar. Se define una medida de similitud s_{ij} (o una distancia d_{ij}) que es calculada entre cada par de objetos. En este grafo cada vértice v_i corresponderá a un objeto y el peso w_{ij} de cada enlace a la similitud (o distancia) entre el par de objetos i y j conectados. Mientras más similares sean dos objetos, mayor es s_{ij} (y menor d_{ij}). Hay tres grafos de similitud que son usualmente utilizados para clustering espectral [38]:

- **Grafo ϵ-vecindario**: aquel grafo que contiene solamente los enlaces de los objetos a una distancia menor a ϵ. Debido a que las distancias entre los puntos son similares entre sí (a lo mucho ϵ), mantener los pesos de los enlaces no incorpora más información. Por

tanto, este grafo es usualmente considerado como no pesado [38].

- **Grafo de k-vecinos más cercanos**: en este tipo de grafo cada vértice v_i se conecta con el vértice v_j, si es que v_j esta entre los k vecinos más cercanos. Para evitar que el grafo sea dirigido existen dos técnicas: ignorar la dirección considerando cada enlace como no dirigido o solo conectar un par de vértices si la conexión anterior se da en ambas direcciones. Este segundo caso es llamado el *grafo mutuo de k-vecinos más cercanos*. En cualquier caso, los pesos de los enlaces corresponden a la similitud de los objetos.

- **Grafo completo**: en que se conectan todos los puntos con similitud positiva y se pesan todos los enlaces usando s_{ij}. Como el grafo debería representar relaciones locales de vecindario, esta construcción es solo útil si la función de similitud representa vecindarios locales. Una medida de similitud normalmente utilizada es $s(x_i, x_j) = \exp(-\|x_i - x_j\|^2/(2\sigma^2))$, siendo σ un parámetro que controla el ancho de los vecindarios.

Independiente del grafo de similitud, dos matrices son utilizadas para el cálculo de la matriz Laplaciana:

- La matriz de adyacencia simétrica W con cada entrada W_{ij} correspondiendo al peso del enlace w_{ij} del grafo de similitud.

- La matriz diagonal de grado D donde cada entrada es definida como $D_{ii} = d_i = \sum_{j=1}^{n} w_{ij}$.

Se pueden identificar tres matrices Laplacianas del grafo:

- **Matriz Laplaciana no normalizada**: $L = D - W$

- **Matriz Laplaciana normalizada simétrica**: $L_{sym} := D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$

- **Matriz Laplaciana normalizada** *random walk*: $L_{rw} := D^{-1} L = I - D^{-1} W$

Cada matriz Laplaciana entrega una variante de Clustering Espectral distinta, siendo aquel que hace uso de la Matriz Laplaciana no normalizada el Clustering Espectral no normalizado.

Siendo k el número de clústeres y S la matriz de similitud, el algoritmo de **Clustering Espectral** no normalizado se muestra en el Algoritmo 3 [38].

Algoritmo 3 Algoritmo de clustering espectral no normalizado

1. Construir un grafo de similitud, siendo W su matriz de adyacencia con los pesos.
2. Computar la matriz Laplaciana no normalizada L.
3. Computar los primeros k vectores propios $u_1, \ldots, u_k$ de L.
4. Sea U la matriz que contiene los vectores $u_1, \ldots, u_k$ como columnas.
5. Para cada $i = 1, \ldots, n$, sea y_i el vector correspondiente a la i-esima fila de U.
6. Agrupar los puntos y_i con el algoritmo *k-means* en los clústeres $C_1, \ldots, C_k$.
7. Retornar los clústeres $A_1, \ldots, A_k$ con $A_i = \{j \mid y_j \in C_i\}$

3.1.5. Clustering basado en modelo

Una posible generalización es suponer que el conjunto de los k centroides de los clústeres son un modelo desde los que se generan los datos. Para generar un objeto en este modelo, se elige un centroide al azar y se agrega un ruido aleatorio a la información. Si el ruido se distribuye normalmente, entonces se obtienen clústeres con forma esférica.

El **C**lustering basado en modelo asume que los datos fueron generados a partir de algún cierto modelo probabilístico, como el descrito anteriormente, e intenta recuperar dicho modelo a partir de los datos [35]. El modelo que se recupera desde los datos define los clústeres y una asignación de los objetos a los clústeres. Esto otorga una mayor flexibilidad a este tipo de clustering pues los clústeres se pueden adaptar a información disponible sobre la forma del conjunto de datos.

Uno de los modelos más usados e importantes en este tipo de clustering es mezcla de Gaussianas en donde se asume que los datos fueron generados a partir de múltiples distribuciones Gaussianas no esféricas. Considerando a cada objeto $x = (x_1, \ldots, x_n)$. Cada una de estas

distribuciones Gaussianas va a estar definida como:

$$x = \mathcal{N}(\mu, \Sigma) \tag{3.4}$$

Siendo μ la media y Σ la covarianza de la distribución Gaussiana.

Se debe tener en cuenta que no todas las Gaussianas van a abarcar la misma cantidad de objetos o, dicho de otro modo, la probabilidad de que un objeto haya sido generado por una Gaussiana no va a ser absoluta ni necesariamente uniforme. Para ello dentro del modelo de mezcla de Gaussianas un último parámetro es el peso asociado a cada gaussiana π_j con $j \in \{1, \ldots, k\}$ donde k es el número de gaussianas (y el número de clústeres).

La meta es estimar cada uno de los parámetros de este modelo de mezcla de Gaussianas: la media μ_i, la varianza Σ y el peso π_j de cada Gaussiana.

Para ello se puede utilizar el algoritmo de Esperanza-Maximización [39, 40] y operar según lo presentado en el Algoritmo 4. Notar que, en el paso de **Maximización**, dependiendo de las probabilidades calculadas en el paso de **Esperanza**, habrá un cambio en la estimación de los distintos parámetros de la Gaussiana hasta que eventualmente no haya ningún cambio y se alcance la convergencia.

Algoritmo 4 Algoritmo Esperanza-Maximización para estimar un modelo de mezclas gaussianas para clustering

---

1. Asumir parámetros aleatorios para la Gaussiana
2. Mientras no haya convergencia:
   a) **Esperanza**: calcular la probabilidad de que cada elemento sea generado por cada Gaussiana
   b) **Maximización**: recalcular parámetros del modelo en base a las probabilidades de que cada elemento pertenezca
3. Asignar los elementos a la Gaussiana con mayor probabilidad para obtener clústeres

---

3.1.6. Mean-Shift

El algoritmo de clustering *Mean-Shift* [41] está basado en centroides candidatos que se actualizan para ser la media de los puntos en una región dada.

De tenerse un cierto centroide candidato x_i para la iteración t, este es actualizado de acuerdo al vector de desplazamiento de media m presentado en la Ecuación 3.5. Este vector es calculado para cada centroide y apunta a aquella región que maximiza el incremento en la densidad de puntos.

$$x_i^{t+1} = m(x_i^t) \tag{3.5}$$

El vector m es calculado usando la Ecuación 3.6 con $N(x_i)$ el vecindario de elementos que están alrededor de x_i y K algún kernel. Esto efectivamente actualiza un centroide para que sea la media de las muestras dentro de su vecindario.

$$m(x_i) = \frac{\sum_{x_j \in N(x_i)} K(x_j - x_i) x_j}{\sum_{x_j \in N(x_i)} K(x_j - x_i)} \tag{3.6}$$

K puede ser el kernel plano expuesto en la Ecuación 3.7 lo que provoca que $m(x_i)$ se actualice al promedio de los puntos en el vecindario.

$$K(x) = \begin{cases} 1 & \text{si } \|x\| \leq \lambda \\ 0 & \text{si } \|x\| > \lambda \end{cases} \tag{3.7}$$

La actualización de los centroides candidatos se realiza mientras no haya convergencia, es decir, mientras haya un desplazamiento de estos.

Al finalizar, varios de los centroides candidatos son filtrados en una etapa de post procesamiento para eliminar aquellos (casi) duplicados para formar el conjunto final de centroides. Esto permite operar desconociendo el número de clústeres, dependiendo en su lugar del parámetro λ de la Ecuación 3.7 que corresponde al *bandwidth* o ancho de banda.

Una vez finalizado, cada centroide define un clúster y cada elemento es asignado al clúster correspondiente al centroide más cercano.

3.1.7. Affinity Propagation

Affinity Propagation [42] es un algoritmo de clustering basado en ”pasar mensajes” entre los objetos.

Cada objeto pasa mensajes a cada otro objeto en base a su semejanza. Estos mensajes indicarán que tan apto es un objeto para ser ejemplar de otros y corresponden a dos categorías:

1. La responsabilidad $r(i,k)$ que corresponde a la evidencia acumulada que el objeto k debería ser el ejemplar del objeto i, tomando en cuenta otros ejemplares potenciales para i.

2. La disponibilidad $a(i,k)$ que corresponde a la evidencia acumulada que el objeto i debería escoger el objeto k como su ejemplar, considerando los valores de otros objetos que k debería ser un ejemplar.

Por lo que, cada ejemplar es escogido si es que son suficientemente similares a muchos objetos y además son escogidos por otros objetos como representativos.

Al iniciar el algoritmo, los valores de responsabilidad y disponibilidad entre cada par de objetos son inicializadas en cero. Luego, iterativamente los objetos se envían mensajes para actualizar estos valores.

Primero se actualiza la responsabilidad. Siendo $s(i,k)$ una medida de la semejanza entre los objetos i y k, la responsabilidad $r(i,k)$ es actualizada en cada iteración de acuerdo a la Ecuación 3.8.

$$r(i,k) \leftarrow s(i,k) - \max_{k' \neq k} \{a(i,k') + s(i,k')\} \tag{3.8}$$

Seguido de esto, se actualiza la disponibilidad $a(i,k)$ de acuerdo a las Ecuaciones 3.9 (para $i \neq k$) y 3.10 (para $i = k$).

$$a(i,k) \leftarrow \text{mín}\,\{0, r(k,k) + \sum_{i' \notin \{i,k\}} \text{máx}\{0, r(i',k)\}\} \tag{3.9}$$

$$a(k,k) \leftarrow \sum_{i' \notin \{k\}} \text{máx}\{0, r(i',k)\} \tag{3.10}$$

Al converger, para cada objeto i, el valor de k que maximicé $a(i,k) + r(i,k)$ identifica k como el ejemplar de i. Hay que recordar que cada ejemplar representará a un subconjunto de los objetos y por tanto a un clúster.

En cada actualización realizada se aplica un factor de *damping* λ para evitar oscilaciones numéricas que aparecen en algunas circunstancias. Para el paso de una iteración t a $t+1$, el factor de *damping* es aplicado a la actualización de las responsabilidades y disponibilidades según lo indicado en las Ecuaciones 3.11 y 3.12, respectivamente.

$$r_{t+1}(i,k) = \lambda \cdot r_t(i,k) + (1-\lambda) \cdot r_{t+1}(i,k) \tag{3.11}$$

$$a_{t+1}(i,k) = \lambda \cdot a_t(i,k) + (1-\lambda) \cdot a_{t+1}(i,k) \tag{3.12}$$

Una ventaja de Affinity Propagation es que el número de ejemplares (i.e. número de clústeres) no debe ser indicado, sino que emerge del procedimiento.

3.1.8. DBSCAN

DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) es un algoritmo de clustering que encuentra objetos núcleo de alta densidad y expande clústeres desde estos. Se ajusta mejor a conjuntos de datos que contienen clústeres de densidad similar.

Este algoritmo se basa en la noción de densidad, en que los clústeres son reconocidos pues la densidad de estos es considerablemente mayor que fuera de este. Además, la densidad en áreas donde hay ruido es menor que aquellas de cualquier clúster.

La idea clave es que cada objeto tendrá un vecindario donde debe haber un número mínimo de objetos, esto es, la densidad del vecindario debe exceder un cierto umbral.

La definición de vecindario para este algoritmo puede ser vista en la Ecuación 3.13 y depende de una medida de distancia $dist(a, b)$ que indica la distancia entre dos objetos a y b, y una magnitud ϵ que define el tamaño del vecindario N_ϵ del punto p. Notar que ϵ corresponde a un parámetro del algoritmo.

$$N_\epsilon(p) = \{q \in S \mid dist(p, q) \leq \epsilon\} \tag{3.13}$$

Con la previa noción de vecindario se pueden definir tres tipos de puntos:

- **Punto de Núcleo**: aquel que se encuentra dentro del clúster. Un objeto p será un Punto de Núcleo si es que el tamaño de su vecindario tiene al menos $minPts$ objetos, esto es, $|N_\epsilon(p)| \geq minPts$.
- **Punto de Borde**: aquel que se encuentra en el borde del clúster. Un objeto p será un Punto de Borde si es que el tamaño de su vecindario tiene menos de $minPts$ objetos, esto es, $|N_\epsilon(p)| < minPts$, pero se encuentra en la vecindad de algún otro punto de núcleo q, es decir, $\exists q \mid p \in N_{eps}(q) \wedge |N_{eps}(q)| \geq minPts$
- **Punto de Ruido**: aquel que se encuentra fuera del clúster. Un objeto p será un Punto de Ruido si es que no es un punto de núcleo ni un punto de borde.

Notar que la cantidad $minPts$ corresponde a un parámetro del algoritmo.

Los clústeres se van a formar tomando en cuenta si los Puntos de núcleo están **densidad-conectados** respecto de ϵ y $minPts$:

1. Un punto p es **directamente densidad-accesible** desde un punto de núcleo q si es que está en el vecindario de este, i.e. $p \in N_\epsilon(q)$.

2. Un punto p es **densidad-accesible** desde un punto q si es que existe una secuencia $p_1, \ldots, p_n$ tal que $p_1 = p$ y $p_n = q$ y además p_{i+1} es directamente densidad-accesible desde p_i para todo $i \in 1, \ldots, n-1$.

3. Un punto p está **densidad-conectado** a un punto q si es que existe un punto o tal que p y q son **densidad-accesible** desde o.

Un clúster se definirá en base a los puntos centrales, expandiéndose desde estos de acuerdo con la densidad-accesibilidad y la densidad-conectividad de los puntos. Si es p es un objeto tal que $p \in C$ y q es densidad-accesible desde p, entonces $q \in C$. Además, $\forall\ p, q \in C$, p estará densidad-conectado a q. Clústeres expandidos a partir de dos o más puntos centrales se irán mezclando en base a estas reglas.

Cabe destacar que un cierto subconjunto de objetos no entrará en los clústeres presentados en el párrafo anterior, esto es, no pertenecerán a ningún clúster y serán clasificados como ruido por el algoritmo.

El número de clústeres entregado por este algoritmo dependerá de los dos hiper-parámetros ϵ y $MinPts$ que definen la densidad de los clústeres. Mientras que $MinPts$ controla principalmente que tan tolerante es el algoritmo hacia el ruido, ϵ debe ser elegido apropiadamente. Si ϵ es muy pequeño, la mayor parte de los datos serán clasificados como ruido. Si es muy alto, causa que clústeres cercanos se mezclen y, que incluso, todos los objetos sean entregados como un solo clúster.

3.1.9. OPTICS

El algoritmo de clustering OPTICS (*Ordering Points to Identify the Clustering Structure*) [43] se asemeja en operación a DBSCAN, pero relaja la restricción de ϵ siendo capaz de ajustar automáticamente el tamaño de cada vecindario. Detecta los clústeres por medio de una jerarquía ordenada por la cercanía a los puntos de núcleo.

De manera similar a DBSCAN, OPTICS operará con dos parámetros: $\mathcal{E}$, el máximo ϵ a considerar al generar el vecindario $N_\epsilon(p)$ de un objeto p y $minPts$ el número mínimo de

puntos que deberán haber en el vecindario de p para considerarse un punto de núcleo.

Sin embargo, OPTICS también generará una grafo de accesibilidad en base a la distancia de núcleo *coreD* (presentada en la Ecuación 3.14) y la distancia de accesibilidad *reachD* (ídem. en 3.15)

$$coreD_{\mathcal{E},minPts}(p) = \begin{cases} \text{Indefinido} & \text{si } |N_{\mathcal{E}}(p)| < minPts \\ \text{La } minPts\text{-era menor distancia en } N_{\mathcal{E}} & \text{en todo otro caso} \end{cases} \tag{3.14}$$

$$reachD_{\mathcal{E},minPts}(q,p) = \begin{cases} \text{Indefinido} & \text{si } |N_{\mathcal{E}}(p)| < minPts \\ \text{máx}(coreD_{\mathcal{E},minPts}(p), dist(p,q)) & \text{en todo otro caso} \end{cases} \tag{3.15}$$

Cabe destacar que si $\mathcal{E}$ es suficientemente grande nunca se dará el caso en que *coreD* o *reachD* quedan con algún valor indefinido, y en tal caso, el vecindario de cada objeto corresponderá a todo el conjunto. Por tanto, el parámetro $\mathcal{E}$ principalmente es requerido para limitar la densidad de clústeres que ya no son de interés y acelerar la ejecución del algoritmo. El parámetro $\mathcal{E}$ no es estrictamente necesario y, por tanto, se puede establecer como $\mathcal{E} = \infty$.

El algoritmo OPTICS producirá un ordenamiento particular de los objetos quedando en posiciones cercanas aquellos objetos que estén cerca entre sí en el espacio, anotados junto a su menor distancia de accesibilidad *reachD*.

Luego, se pueden usar métodos visuales y automáticos para detectar los clústeres [43, 44], detectando variaciones del valor anotado de *reachD* al avanzar a través del ordenamiento. Al recorrer los puntos en el ordenamiento, los valores de *reachD* formaran valles y picos, donde cada valle corresponderá a un clúster y los picos las separaciones entre los valles, usualmente conteniendo puntos de ruido.

Un ejemplo de esto puede ser observado en la Figura 3.1. En la esquina superior izquierda se encuentran los objetos a agrupar: puntos en el plano 2D. En la esquina superior derecha se muestra un dendrograma mostrando la jerarquía de objetos producto del ordenamiento de OPTICS y los clústeres detectados denotados con los colores azul, rojo y verde. En el gráfico

de la parte inferior, se muestran en el eje Y los valores de accesibilidad *reachD* de los objetos ordenados en el eje X. Notar como los valles se corresponden con cada clúster. La porción de los picos, de color amarillo, representa puntos de ruido que se encuentran rodeando los clústeres en áreas de baja densidad.

Figura 3.1: Ejemplo gráfico de algoritmo OPTICS ($\mathcal{E}$ = 0.5 y $minPts$ = 10).

Cabe destacar nuevamente que OPTICS permite detectar clústeres a pesar de que la densidad de los puntos dentro de estos puede ser variable, pero tendrá una sensibilidad similar a la del algoritmo DBSCAN respecto del parámetro $\mathcal{E}$. Para valores bajos habrá tendencia a clasificar gran parte de los datos como ruido.

3.2. Medidas de similitud

Aunque los algoritmos de clustering cumplen el mismo propósito, tienen claras diferencias en como determinan los clústeres para un conjunto de datos.

No existe de manera intrínseca un algoritmo de clustering que sea mejor que otro, sin embargo, las salidas producidas por dos algoritmos de clustering distintos podrían ser comparadas

y observar que tan similares son los conjuntos de clustering generados.

En base a esta comparación se podría establecer si vale la pena usar un algoritmo más sofisticado y que entregue más información respecto al conjunto de datos, o si es que basta con alguno más sencillo y eficiente.

Para este propósito se han considerado dos métricas previas: el **Índice Rand** y el **Índice Rand Ajustado** que han sido ampliamente utilizadas en estudios con comparación de clústeres [45]. En este trabajo se propone una nueva técnica que comparará la salida por medio de los centroides que es el **Índice de Emparejamiento Único**.

3.2.1. Índice Rand

El **Índice Rand** (RI, del inglés *Rand Index*) [46] es una medida de similitud de la salida de dos algoritmos de clustering. Se basa en contar la cantidad de acuerdos entre las salidas de algoritmos de clustering.

Un acuerdo ocurre si es que, para un par de objetos del conjunto de datos, ambos algoritmos clasifican estos objetos en el mismo clúster o en clústeres distintos.

Definición 3.3 (Índice Rand). Dado un conjunto de ubicaciones S y dos conjuntos de clústeres A y B generados por algoritmos de clustering α y β respectivamente. Se definirá lo siguiente:

- Sea a el número de pares de elementos de S que están en el mismo clúster en A y en el mismo clúster en B.
- Sea b el número de pares de elementos de S que están en distintos clústeres en A y distintos clústeres en B.
- Sea c el número de pares de elementos de S que están en el mismo clúster en A, pero en diferentes clústeres en B.

- Sea d el número de pares de elementos en S que están en distintos clústeres en A, pero en el mismo clúster en B.

Donde $a + b$ corresponde al número de acuerdos, mientras que $c + d$ es el número de desacuerdos.

El **Índice Rand** es definido como:

$$RI(A, B) = \frac{a + b}{a + b + c + d} \tag{3.16}$$

El RI toma un valor entre 0 y 1, con 0 indicando que ambos algoritmos de clustering no están de acuerdo para ningún par de puntos y 1 indicando que ambos algoritmos generaron un clustering exactamente igual.

Un problema del RI es que en la práctica el rango de valores que toma esta entre 0.5 y 1, es decir, el valor de esta medida esta inflado y puede entregar valores altos aun cuando la asignación de clústeres se realiza aleatoriamente [47, 45].

3.2.2. Índice Rand Ajustado

El **Índice Rand Ajustado** (ARI, del inglés *Adjusted Rand Index*) [48, 47] es la versión corregida para aleatoriedad del RI. La corrección se realiza estableciendo como base la similitud esperada de todas las comparaciones entre pares como especificaría un modelo aleatorio. La definición de ARI puede ser vista en la Definición 3.4.

Definición 3.4 (Índice Rand Ajustado). Sean RI una función que entrega el Índice Rand, A y B dos conjuntos de clustering generados por algoritmos de clustering α y β respectivamente, y E la esperanza. Se define el Índice Rand Ajustado entre los conjuntos A y B como $ARI(A, B)$.

Su valor corresponde al siguiente ajuste:

$$ARI(A, B) = \frac{RI(A, B) - E[RI]}{\text{máx}(RI) - E[RI]} \tag{3.17}$$

Usando los valores de a, b, c y d de la Definición 3.3 se puede calcular como [47, 49]:

$$ARI(A, B) = \frac{2(ab - cd)}{(b + c)(c + d) + (b + d)(d + a)} \tag{3.18}$$

El ARI puede entregar un valor entre -1 y 1. Al igual que el RI, un valor de ARI igual a 1 implica que ambos algoritmos generaron un conjunto de clústeres exactamente igual. El valor de ARI se reduce a medida que los algoritmos de clustering van generando un mayor número de desacuerdos entre sí, y cuando llega a 0 es indicio que las salidas fueron tan radicalmente distintas que es similar a haber generado los clústeres aleatoriamente. Valores menores a cero indican diferencias en los modelos de clustering, por ejemplo, en los parámetros utilizados o en la implementación.

Ejemplo 3.1. Sea $S = \{s_1, \ldots, s_6\}$ un conjunto de ubicaciones. Definiendo algunos posibles conjuntos de clusters:

- $A_1 = \{\{s_1, s_2\}, \{s_3, s_4\}, \{s_5, s_6\}\}$
- $A_2 = \{\{s_5, s_6\}, \{s_1, s_2\}, \{s_3, s_4\}\}$
- $A_3 = \{\{s_4, s_5, s_6\}, \{s_1, s_2\}, \{s_3\}\}$
- $A_4 = \{\{s_5\}, \{s_1\}, \{s_2, s_3, s_4, s_6\}\}$
- $A_5 = \{\{s_1, s_3\}, \{s_5, s_6\}, \{s_2, s_4\}\}$

Notar que el valor de ARI será mayor mientras más acuerdos haya entre conjuntos de clústeres. De esta manera:

1. El valor $ARI(A_1, A_2) = +1.00$ porque estos clústeres son un permutación de los mismos subconjuntos. Por lo que hay un calce perfecto cuyo valor de ARI es el máximo.

2. En el conjunto A_3, el objeto s_4 cambia a un clúster distinto. Por tanto, el valor de ARI decrece (a $ARI(A_1, A_3) = +0.44$) junto al número de acuerdos.

3. Los conjuntos A_4 y A_5 son poco similares obteniéndose para este caso un valor cercano a cero ($ARI(A_4, A_5) = -0.06$). Estos conjuntos de clústeres se generaron aleatoriamente para este ejemplo.

3.2.3. Índice de Emparejamiento Único

Dos algoritmos de clustering al generar los clústeres en elementos esparcidos homogéneamente en el espacio podrían generar diferencias menores en los bordes de los clústeres que llevan a una penalización en las medidas vistas en las secciones anteriores.

En estos casos, esto provocará que los centroides de los clústeres se muevan muy poco en el espacio. Es deseable el comparar las posiciones de los clústeres.

El **Índice de Emparejamiento Único** (UMI, del inglés *Unique Match Index*) correspondiente a la proporción de emparejamientos únicos entre conjuntos de centroides. Un emparejamiento único ocurre cuando solamente un centroide del segundo conjunto tiene un cierto centroide del primer conjunto como el más cercano, tal cual lo indica la Definición 3.5 de este.

Definición 3.5 (Índice de Emparejamiento Único). Sean $A = \{a_1, \ldots, a_n\}$ y $B = \{b_1, \ldots, b_n\}$ dos conjuntos de clústeres cada uno con conjuntos de centroides $C_A = \{c_{a_1}, \ldots, c_{a_n}\}$ y $C_B = \{c_{b_1}, \ldots, c_{b_n}\}$, respectivamente. Sea $dist(x, y)$ una función que entrega la distancia entre las ubicaciones o centroides x e y.

Sea también $M(c, C_B) \subset C_B$ un subconjunto de centroides cuyos elementos tienen al centroide $c \in C_A$ como el centroide más cercano. Así, $|M(c, C_B)|$ es el número de parejas de c en C_B

$$M(c, C_B) = \left\{ z \;\middle|\; z \in C_B, dist(z, c) < dist(z, c') \; \forall c' \in C_A, c' \neq c \right\} \tag{3.19}$$

Por tanto, se define el **Índice de Emparejamiento Único** de los centroides en C_A respecto de aquellos en C_B como $UMI(C_A, C_B)$ y corresponde a los centroides en C_A con una única pareja en C_B.

$$UMI(C_A, C_B) = \frac{\left| \left\{ c \;\middle|\; c \in C_A, \; |M(c, C_B)| = 1 \right\} \right|}{n} \tag{3.20}$$

Un ejemplo gráfico de cómo opera el UMI puede ser visto en la Figura 3.2 y como este cambia al variar las posiciones de los centroides:

Ejemplo 3.2. Dados dos conjuntos de clústeres $A = \{a_1, a_2, a_3, a_4\}$ y $B = \{b_1, b_2, b_3, b_4\}$ con 4 clústeres cada uno, con sus conjuntos de centroides siendo $C_A = \{c_{a_1}, c_{a_2}, c_{a_3}, c_{a_4}\}$ y $C_B = \{c_{b_1}, c_{b_2}, c_{b_3}, c_{b_4}\}$, respectivamente. Como se muestra en la Figura 3.2, se pueden destacar dos escenarios diferentes.

- En la Figura 3.2a se muestra un calce completo entre los conjuntos de centroide. Cada centroide tiene una única pareja en ambos conjuntos dada por pequeñas diferencias en la salida del algoritmo de clustering.
- En el escenario opuesto, mostrado en la Figura 3.2b, los centroides C_{a_2} y C_{a_3} se alejan de sus parejas previas, acercándose a otros centroides en C_B que ya estaban emparejados. Aquí se muestra que UMI no es una medida simétrica y su valor puede ser igual a cero.

$UMI(C_A, C_B) = 4/4 = +1.00$
$UMI(C_B, C_A) = 4/4 = +1.00$

(a) Calce completo.

$UMI(C_A, C_B) = 2/4 = +0.50$
$UMI(C_B, C_A) = 0/4 = +0.00$

(b) Calce parcial.

Figura 3.2: Ejemplo de Índice de Emparejamiento Único. Los centroides de dos conjuntos A y B se representan por los cuadrados negros y blancos respectivamente. Las flechas apuntan al centroide más cercano del otro conjunto. Los centroides con una sola flecha entrante tienen una única pareja e incrementan el valor del UMI.

Capítulo 4

Framework para la creación de instancias

En este capítulo se presenta el framework de creación de instancias para permitir la relajación de escenarios reales complejos o instancias de gran tamaño.

El framework consta de varias transformaciones de los datos originales. Un diagrama mostrando las diferentes transformaciones involucradas puede ser visto en la Figura 4.1.

Figura 4.1: Diagrama de flujo de datos del framework de creación de instancias.

El framework trabaja asumiendo que existen cuatro fuentes de datos:

1. Una Red de Caminos.
2. Ubicaciones geográficas que pueden representar paraderos, terminales o sitios de interés.
3. Zonas geográficas dividiendo la ciudad y agrupando los paraderos.
4. Una Matriz de Demanda Zonal Temporal, que indica el número de pasajeros que desean desplazarse entre cada zona en cada ventana de tiempo.

La Definición 4.1 presenta la definición de una zona. Se puede observar que es muy similar a la Definición 3.2 para clústeres. La diferencia radica en su origen: un clúster es generado por un algoritmo de clustering, mientras que una zona es parte de los datos originales y es establecida por el diseñador de la red.

Definición 4.1 (Zonas). Dado un conjunto de ubicaciones S, un **conjunto de zonas** Z incluye k subconjuntos de ubicaciones.

$$Z = \left\{ z_i \;\middle|\; z_i \subset S \;\wedge\; z_i \cap z_j = \emptyset \;\forall i \neq j \;\wedge\; \bigcup_{1 \leq i \leq k} z_i = S \right\} \tag{4.1}$$

Así, cada **zona** $z_i \in Z$, esto es, cada conjunto de ubicaciones, tiene su propio centroide C_{z_i}. Desde aquí, usamos la notación reducida $z \in Z$ para cada clúster; por tanto, se entiende que $|Z| = k$.

El framework de creación de instancias produce versiones relajadas de los dos componentes principales de una instancia del UTRP: la **Red de Caminos** y la **Matriz de Demanda**. En las secciones siguientes se muestra cómo se generan estas componentes a partir de los datos.

4.1. Red de Caminos Relajada

La **Red de Caminos Relajada** (**RRN**, del inglés ***Relaxed Road Network***) es generada a partir de los clústeres (que agrupan las ubicaciones) y la red de caminos original como se puede observar en la Figura 4.1.

Para ello se aplica el siguiente criterio: *Los centroides de dos clústeres se conectan, si y solo si hay un par de ubicaciones, una de cada clúster, conectadas en la red de caminos original*.

Una RRN básica se muestra en el Ejemplo 4.1, mientras que la Definición 4.2 formaliza este concepto.

Definición 4.2 (Red de Caminos Relajada). Dada una red de caminos $R = (N, L)$ y un conjunto de clústeres A generados por un algoritmo de clustering α. Entonces $\hat{N}$ es el conjunto de centroides generados por α:

$$\hat{N} = \{C_a \mid a \in A\} \tag{4.2}$$

y $\hat{L}$ es un conjunto de enlaces que conectan los centroides de $\hat{N}$:

$$\hat{L} = \{(C_a, C_{a^*}) \mid \exists (n_i, n_j) \in L \wedge n_i \in a \wedge n_j \in a^*\} \tag{4.3}$$

Así, sea $\hat{R}$ una **RRN** representada por un grafo no dirigido tal que $\hat{R} = (\hat{N}, \hat{L})$.

Ejemplo 4.1. Sea $R = (N, L)$ una red de caminos con $|N| = 16$ ubicaciones y sus conjunto de conexiones L, como se muestra en la Figura 4.2. Un algoritmo de clustering α agrupa las ubicaciones en el conjunto de clústeres $A = \{a_1, a_2, a_3\}$. El centroide C_{a_2} se conecta a los otros centroides pues hay conexiones entre al menos una de sus ubicaciones en R. Los centroides C_{a_1} y C_{a_3} se encuentran desconectados porque no hay ubicaciones en R conectadas entre sí que pertenezcan a los clústeres correspondientes.

Figura 4.2: Ejemplo de construcción de RRN. Los círculos representan ubicaciones dentro de los clústeres y los cuadrados los centroides de cada clúster. La Red de Caminos resultante conecta el centroide blanco (C_{a_2}) con los centroides negro (C_{a_1}) y gris (C_{a_3}). Estos enlaces son generados porque hay al menos una ubicación directamente conectada entre cada par de clústeres.

4.2. Matriz de Demanda Relajada

La información de demanda de la instancia reducida es representada por una matriz con doble peso llamada **Matriz de Demanda Relajada** (**RDM**, del inglés ***Relaxed Demand Matrix***). La RDM representa la demanda del escenario original para la nueva instancia reducida. La matriz se genera como sigue:

1. Calcular el **Peso basado en Ubicación** de cada clúster respecto a cada zona
2. Calcular el **Peso basado en Distancia** de cada clúster respecto a cada zona
3. Combinarlos para obtener el **Factor de Doble Peso** para la corrección de demanda
4. Reducir la dimensión de tiempo de la matriz de demanda zonal, pues el UTRP usa información de demanda constante
5. Calcular la matriz de demanda relajada combinando los resultados anteriores.

4.2.1. Peso basado en Ubicación

Las ubicaciones de una zona pueden distribuirse entre muchos clústeres, así la demanda de cada zona debe ser distribuida proporcionalmente entre cada clúster involucrado. Se define el *Peso basado en Ubicación* como la proporción de ubicaciones de una zona en un clúster respecto al total de ubicaciones en dicha zona.

Definición 4.3 (Peso basado en Ubicación). Sea A un conjunto de clústeres y Z un conjunto de zonas, donde cada elemento $a \in A$ y cada elemento $z \in Z$ son conjuntos de ubicaciones. Por cada par de clúster-zona (a, z), el peso basado en ubicación, $prop_{az}$, es la proporción de ubicaciones tanto en la zona z como en el clúster z respecto del total de ubicaciones en la zona z.

$$prop_{az} = \frac{|a \cap z|}{|z|} \tag{4.4}$$

4.2.2. Peso basado en Distancia

La mayoría de los clústeres pueden cubrir una amplía área. Por lo que sus centroides son adecuados para representar la demanda a ubicaciones cercanas, pero a medida que la distancia aumenta, el centroide se considera se vuelve más impreciso para estas ubicaciones alejadas. Por tanto, definimos el **Peso basado en Distancia** para cada centroide de clúster.

Definición 4.4 (Peso basado en Distancia). Sea A un conjunto de clústeres y Z un conjunto de zonas, donde cada elemento $a \in A$ y cada elemento $z \in Z$ son conjuntos de ubicaciones. Por cada par de clúster-zona (a, z). Se define $dist(a, z)$ como la distancia entre dos ubicaciones o centroides.

Entonces, el Factor de Lejanía, dp_{az}, es la proporción de distancia a un parámetro de distancia

máxima $dist_{max}$. De esta manera, es posible chequear si el centroide de una zona está dentro de una circunferencia centrada en el centroide del clúster con radio $dist_{max}$. Hay que notar que $dist_{max}$ es un parámetro establecido por el diseñador de la red.

$$dp_{az} = \frac{dist_{az}}{dist_{max}} \tag{4.5}$$

Por último, el **Peso basado en Distancia**, df_{az}, es un factor lineal entre [0, 1] que incrementa la contribución de demanda de una zona a un clúster mientras más cerca se encuentren entre sí.

$$df_{az} = \text{máx}\left\{1 - dp_{az}, 0\right\} \tag{4.6}$$

4.2.3. Factor de Doble Peso

Las definiciones previas se combinan en un **Factor de Doble Peso** que será incorporado en el posterior cálculo de la RDM. Este factor se compone del producto del *Peso basado en Distancia* y el *Peso basado en Ubicación* tal como lo muestra la Definición 4.5, y se calcula para cada combinación de centroides de clúster y de zona.

Definición 4.5 (Factor de Doble Peso). Dado un conjunto de clústeres A y un conjunto de zonas Z. Para cada clúster $a \in A$, su **Factor de doble peso** respecto a una zona $z \in Z$, es definido como la multiplicación del *peso basado en distancia* y el *peso basado en ubicación*.

$$dw(a, z) = df_{az} \cdot prop_{az} \tag{4.7}$$

Expandiendo cada término, considerando sus definiciones expuestas con anterioridad, se obtiene que:

$$dw(a,z) = \underbrace{\text{máx}\left\{1 - \frac{dist(C_a, C_z)}{dist_{max}}, 0\right\}}_{\text{peso basado en distancia}} \quad \cdot \quad \underbrace{\left(\frac{|a \cap z|}{|z|}\right)}_{\text{peso basado en ubicación}} \tag{4.8}$$

Por consiguiente, las ubicaciones y las zonas cuya influencia de demanda es marginal; es decir, están suficientemente lejos del centroide del clúster; son tratadas como *outliers*, como puede ser visto en el Ejemplo 4.2.

Ejemplo 4.2. Sea $\hat{R}$ una RRN y $Z = \{z_1, z_2\}$ un conjunto de zonas como se muestra en la Figura 4.3. Hay que notar que $\hat{R}$ está compuesto por un conjunto de centroide $\hat{N} = \{C_{a_1}, C_{a_2}\}$ asociado a un conjunto de clústeres $A = \{a_1, a_2\}$. Para los propósitos de este ejemplo, no es necesario mostrar los enlaces que conectan la RRN.

Figura 4.3: Ejemplo de Factor de doble peso. Los círculos representan ubicaciones: los elementos de a_1 y a_2 están coloreados negro y blanco, respectivamente. El círculo gris centrado en C_{a_1} (cuadrado negro) con radio $dist_{max}$ acota la contribución de demanda de aquellas zonas que intersectan al clúster. Notar que, cada centroide de zona está representado por una estrella.

Notar como cada zona contribuye un diferente número de elementos, o sea, de demanda a cada centroide de los clústeres. Por lo que:

1. La contribución de demanda de la zona z_1 al clúster a_1 se corrige por un peso basado

en distancia $\in [0, 1]$, porque su distancia hacia el centroide C_{a_1} es inferior a $dist_{max}$. Mientras tanto, el peso basado en distancia es igual a $\frac{|a_1 \cap z_1|}{|z_1|} = \frac{6}{6} = 1$, porque el clúster a_1 contiene todas las ubicaciones de la zona z_1. Hay que notar que la ubicación l_1 no es un outlier, pues pertenece a a_1.

2. La zona z_2 no contribuye demanda al clúster a_1, esto pues el peso basado en distancia es igual a cero, puesto que C_{z_2} se encuentra fuera del borde delimitado por la distancia $dist_{max}$ desde C_{a_1}. Esto es a pesar del valor positivo de su peso basado en ubicación: $\frac{|a_1 \cap z_2|}{|z_2|} = \frac{1}{9}$. Notar que, la ubicación l_2 es un outlier en la asignación de clústeres.

3. La zona z_2 contribuye una demanda proporcional al clúster a_2, porque $\frac{|a_2 \cap z_2|}{|z_2} = \frac{8}{9} < 1$. Por sencillez, se obvia el efecto del peso basado en distancia.

4.2.4. Reducción de dimensión temporal en datos de demanda

Los datos de demanda originales incluyen la información de viajes entre cada par de zonas en un itinerario de tiempo dividido en varias ventanas de tiempo en la Matriz de Demanda Zonal Temporal como fue indicado en la Figura 4.1. Por cada ventana de tiempo, hay una cantidad diferente de viajes entre las zonas introduciendo una nueva dimensión a la matriz de demanda. Sin embargo, el UTRP establece una matriz de demanda en una ventana de tiempo única. Por tanto, la dimensión temporal necesita ser reducida a una sola ventana de tiempo.

Para ello la Matriz de Demanda Zonal Temporal es simplificada a una **Matriz de Demanda Zonal Simple** tal como lo indica la Definición 4.6.

Definición 4.6 (Matriz de Demanda Zonal Simple). Dado un conjunto de zonas $Z = \{z_1, \ldots, z_k\}$, un conjunto de ventanas de tiempo T, y una **Matriz de Demanda por Ventanas de Tiempo** no simétrica $F_{|Z| \times |Z| \times |T|}$. Cada entrada F_{xyw} de esta matriz corresponde al número de viajes entre las zonas z_x y z_y durante el transcurso de la ventana de tiempo t_w. Hay que notar que F es una matriz hueca, o sea, $F_{xxw} = 0 \ \forall x \in \{1, \ldots, k\}$.

Para cada par de zonas $(z_x, z_y) \in Z^2$, se elige la ventana de tiempo con la máxima cantidad de viajes.

$$t' = \arg\max_{t \in T}(F_{xyt}) \tag{4.9}$$

Así la **Matriz de Demanda Zonal Simple** $E_{|Z|\times|Z|}$ es simétrica y cada entrada denota la cantidad máxima de viajes entre las dos zonas para todas las ventanas de tiempo.

$$E_{xy} = \text{máx}\{F_{xyt'}, F_{yxt'}\} \tag{4.10}$$

4.2.5. Agregación de resultados

Una vez que todos estos elementos están definidos, se combinan según la Definición 4.7 para generar la **RDM**.

Definición 4.7 (Matriz de Demanda Relajada). Dada una RRN $\hat{R}$ compuesta de un conjunto de centroides $\hat{N} = \{C_{a_1}, \ldots, C_{a_m}\}$ asociada a un conjunto de clústeres $A = \{a_1, \ldots, a_m\}$, y un conjunto de zonas $Z = \{z_1, \ldots, z_k\}$ y una *Matriz de Demanda Zonal Simple* $E_{|Z|\times|Z|}$. La **Matriz de Demanda Relajada** $D_{|A|\times|A|}$ contiene la cantidad de viajes entre cualquier par de nodos $(C_{a_x}, C_{a_y}) \in \hat{N}^2$

$$D_{xy} = \sum_{z_i \in Z} \sum_{z_j \in Z} dw(a_x, z_i) \times dw(a_y, z_j) \times E_{ij} \ \forall (a_x, a_y) \in A^2 \tag{4.11}$$

4.3. Parámetros de las rutas

El ajuste de los parámetros de las rutas no es una parte central del proceso de creación de instancias y puede ser visto como una parte del problema de diseño de la red y en algunos

modelos es otro aspecto que debe ser optimizado.

De todas formas, se presenta un método que se basa en las distancias de la RRN. Los largos mínimos y máximos de cada ruta se basan en el largo real de una ruta de bus, dividiendo este valor por la distancia promedio entre los nodos de la red de caminos generada.

La cantidad de rutas fue asignada a partir de pruebas usando un algoritmo de construcción aleatorio similar al descrito en [8] pero usando un largo mínimo de dos nodos para cada ruta. Múltiples valores son probados, eligiendo aquel con el que el algoritmo de construcción pudiese generar un conjunto de rutas factible con una tasa de éxito arbitraria.

Capítulo 5

Experimentación

5.1. Configuración experimental

En este capítulo, se aplica el framework de creación de instancia al caso de estudio seleccionado: la red de buses de RED del sistema de transporte público de Santiago de Chile. La información del sistema de RED es publicada anualmente por el DTPM (Directorio de Transporte Público Metropolitano) e incluye las ubicaciones de los paraderos y la demanda de los pasajeros agrupadas por zonas [50].

5.1.1. Algoritmos de Clustering

Un componente central del framework es la generación de clústeres para poder obtener la RRN y RDM de la instancia reducida.

No obstante, como se vio en el Capítulo 3, existen múltiples algoritmos de clustering disponibles que se pueden adaptar a distintos casos de uso. Estos algoritmos están disponibles en la biblioteca *scikit-learn* [49]. Así se probaron todos los algoritmos presentados anteriormente.

La diferencia de rendimiento de los algoritmos de clustering seleccionados es observable midiendo la similitud de sus salidas lo que permitirá detectar fuertes variaciones que pudiesen

aparecer durante el proceso de relajación.

Hay que notar que la mayoría de los algoritmos seleccionados son estocásticos, por lo que la comparación se realiza sobre 10 ejecuciones de cada algoritmo. Las siguientes medidas de similitud, presentadas en la Sección 3.2, son utilizadas para esta tarea: (1) el Índice de Rand Ajustado (ARI) y (2) el Índice de Emparejamiento Único (UMI).

Cabe recordar que, para ambas medidas, valores más cercanos a su máximo implica una mayor similitud entre las salidas de los algoritmos de clustering.

5.1.2. Red de caminos relajada

La información del sistema RED incluye la posición de 11.320 paraderos y 803 zonas de demanda de pasajeros. La posición de cada paradero está asociada a un par de coordenadas geográficas que incluye su latitud y longitud. Todos los paraderos están dispuestos de a pares tomando en cuenta el sentido bidireccional del tráfico. Lamentablemente, no se incluye información sobre los enlaces entre los paraderos.

Considerando todo lo anterior, antes que la RRN sea generada, los datos de RED son preprocesados como sigue:

1. La posición de cada paradero es transformada a una sistema de coordenadas Euclidiana en 3D asumiendo que el planeta Tierra tiene forma esférica con radio igual a 6371 km aplicando la Definición 5.1. De esta forma, la distancia Euclidiana es usada entre cualquier par de ubicaciones.

2. Cada par de paraderos deben ser mezclados puesto que el UTRP asume paradas bidireccionales en sus rutas. Un algoritmo de Clustering Aglomerativo [35] es usado enlazando clústeres que se encuentren a una distancia de 200 metros o menos. Cada centroide generado se vuelve un paradero único que cubre un área de mayor tamaño. Este procedimiento reduce el número de ubicaciones en un 62.3 %, desde 11.320 a 4.260 nodos.

3. La red de caminos usada por el framework es construida a partir de los nodos del paso anterior enlazados por medio de un algoritmo k-neighbors [35], usando $k = 7$ como el valor del principal parámetro. En este paso, una red completamente conectada es generada, evitando enlaces que atraviesen ciertas extensiones inválidas de terreno, como, por ejemplo, colinas, parques grandes, aeropuertos, etc. Sin embargo, algunos enlaces no toman en cuenta características urbanas más pequeñas, por ejemplo, pasando por encima de ríos ignorando si es que hay puentes cercanos. No hay forma de evitar este problema sin el apoyo de datos geográficos.

4. La distancia entre nodos fue transformada a tiempo en minutos asumiendo que los buses tienen una velocidad promedio de 19.31 km/h basado en reportes previos [51].

Definición 5.1 (Transformación de coordenadas geográficas). Sean θ la latitud y ϕ la longitud de una ubicación l. Asumiendo el planeta Tierra como una esfera de radio $R = 6.371.000$, se define $pos(l)$ como la posición de l como un vector de coordenadas Euclidianas en 3D.

$$pos(l) = [x, y, z]^T = [R\cos(\theta)\cos(\phi), R\cos(\theta)\sin(\phi), R\sin(\phi)]^T \tag{5.1}$$

Para poder generar múltiples instancias reducidas diferentes versiones de la RRN son creadas. Esta manera, para cada algoritmo de clustering seleccionado, tres tamaños de RRN fueron definidos: 35, 90 y 135 clústeres.

5.1.3. Matriz de demanda relajada

El DTPM publica la información de demanda de pasajeros como matrices que incluyen el número promedio de viajes entre cada par de las 803 zonas predefinidas. Se calcula el promedio sobre cinco días laborales y es presentado para un período de 24 horas dividido en 48 ventana de tiempo de media hora cada una. Estas matrices fueron generadas rastreando

el comportamiento de los usuarios desde las bases de datos de la *smart-card* Tarjeta bip! e información de GPS de los buses [50].

Recordar que en la Sección 4.2 se requería ajustar el parámetro $dist_{max}$ presente en la Definición 4.5 que acotaba el factor basado en distancia del factor de doble peso. Este parámetro se estableció como la distancia promedio entre todos los centroides conectados en la RRN, así como lo indica la Ecuación 5.2.

$$dist_{max} = \overline{dist} = \frac{1}{|\hat{L}|} \sum_{(C_i, C_j) \in \hat{L}} dist(C_i, C_j) \tag{5.2}$$

5.1.4. Parámetros de las rutas

Respecto a los parámetros de las rutas, se consideró que la ruta de buses más larga mide 97 km (basado en el reporte [52]). El largo mínimo se calculó de manera similar, pero asumiendo un décimo del largo previamente indicado, esto es 9,7 km.

Hay que recordar que la cantidad de rutas se establece observando la tasa de éxito de un algoritmo de construcción aleatoria. Iterativamente se incrementa el número de rutas hasta que dicho algoritmo tuviese una tasa de éxito de un 85 % en generar un conjunto de rutas factible. Sin embargo, debido al pobre rendimiento detectado del algoritmo, se usa un largo mínimo de 2 nodos en lugar del verdadero valor de la instancia. Notar que, es más difícil obtener un conjunto de rutas factibles con una cantidad menor de rutas.

5.1.5. Ambiente de experimentación

La aplicación del framework a los datos de RED se encuentra resumida en la Figura 5.1.

El framework y los experimentos de este trabajo fueron implementados en un Jupyter Notebook ejecutando Python 3.8.1 como el kernel. Sin embargo, el algoritmo de construcción de [8] usado para ajustar los parámetros de las rutas fue implementado en C++ y compilado usando gcc 8.1.0 MinGW-W64.

Figura 5.1: Diagrama de flujo de datos del framework de creación de instancias aplicado a los datos de RED.

Las pruebas fueron ejecutadas en una estación de trabajo con un CPU AMD FX-8350 4.0GHz, 16GB de memoria RAM y el sistema operativo Windows 10 Pro 64-bit.

5.2. Resultados y Discusión

En esta sección se discuten los resultados de los diferentes experimentos para el framework.

Se comienza con una discusión respecto a las capacidades de relajación de los distintos algoritmos de clustering para escoger y recomendar aquellos considerados adecuados para esta tarea.

La sección continúa revisando la facultad del framework de mantener las características del escenario real, revisando su similitud en términos de la estructura de la red de caminos y

la distribución de demanda, en ese orden. Para la distribución de la demanda se presenta una comparación con algunas instancias previas, mostrando la capacidad del framework de generar instancias con características de la vida real.

Por último, se presenta la comparación entre técnicas de clustering para chequear la relevancia de escoger el algoritmo de clustering correcto entre aquellos recomendados.

5.2.1. Recomendación de algoritmos de clustering

Para los experimentos, hay preferencia por aquellos algoritmos que se ajustaban al escenario relajado generando redes completamente conectadas y que eran lo suficientemente flexibles para generar instancias de distinto tamaño.

De los algoritmos de clustering presentados en el Capítulo 3, los que cumplían con estas dos características eran aquellos que permitían indicar el *Número de Clusters* como parámetro del algoritmo. Por tanto, para el resto de los experimentos se seleccionaron los siguientes algoritmos: K-Means, Aglomerativo, Espectral y Mezcla de Gaussianas.

El resto de los algoritmos fueron descartados pues demostraron ser menos versatiles. Mean Shift y Affinity Propagation requerían un cuidadoso ajuste de sus hiper-parámetros para generar distintos escenarios y para algunos valores eran incapaces de generar escenarios válidos. Adicionalmente, DBSCAN y OPTICS eran incapaces de manejar adecuadamente la distribución homogenea de paraderos del caso de estudio, puesto que su procedimiento de reducción de ruido aislaba o obviaba la mayoría de las áreas perifericas.

En la Figura 5.2, un ejemplo de estos comportamientos puede ser visto. Mientras que el algoritmo de Clustering Aglomerativo genera una RRN bien ajustada con 135 nodos, Affinity Propagation necesito un ajuste de su hiper-parámetro *damping* para generar la RRN con 61 nodos presentada siendo incapaz de generar instancias con una menor o mayor densidad de nodos. Por otro lado, DBSCAN genera clusters con grandes diferencias en tamaño que podrían quedar desconectadas. Este comportamiento es notorio en zonas perifericas y es producto del procedimiento de reducción de ruido propio de este algoritmo.

(a) K-Means 135 nodos. **(b)** Affinity Propagation 61 nodos. **(c)** DBSCAN 180 nodos.

Figura 5.2: Comparación de RRN generada por diferentes algoritmos de clustering.

5.2.2. Instancias generadas

A partir de la recomendación anterior y los distintos números de clústeres dado especificados en la Sección 5.1.2, doce nuevas instancias de UTRP fueron generadas. Las características de estas instancias se muestran en la Tabla 5.1.

Es posible ver que las métricas de complejidad de la mayoría de las instancias UTRP generadas son bastante similares para cada escenario de relajación. No obstante, hay una ligera desviación en aquellas instancias en que se aplicó clustering espectral, pues los nodos generados se encontraban más separados, lo que redujo el número de enlaces y esto complico la generación de conjuntos de rutas factibles. De todas formas, estas diferencias pueden ser consideradas menores y no necesariamente indican que instancias producidas usando clustering espectral son más difíciles de resolver u optimizar.

Cada algoritmo recomendado era capaz de correctamente ajustarse a cada escenario de relajación. Tal cual se muestra en la Figura 5.3, mientras mayor es el número de clústeres, más se aproxima la RRN al escenario original. Hay que notar, que la relajación más compleja (Figura 5.3c) tiene uno o dos órdenes de magnitud menos que las ubicaciones mezcladas (Figura 5.3d) y la instancia de RED original, respectivamente.

Al inspeccionar visualmente, la estructura de las instancias es bastante similar con algunas

Tabla 5.1: Características de las instancias de UTRP generadas por el framework aplicando diferentes algoritmos de clustering. La sigla *Agg* corresponde a Aglomerativo, *Kmn* a K-means, *Spc* a Espectral y *Gau* a Mezcla de Gaussianas.

Nombre	Nodos	Enlaces	Distancia prom.	q	$\lambda_{min} - \lambda_{max}$	LB_{pass}	MST
StgoAgg35	35	68	4744.66	8	3 - 21	35.7038	426
StgoKmn35	35	72	4842.07	8	3 - 21	35.7178	439
StgoSpc35	35	64	5388.50	11	3 - 19	36.1139	475
StgoGau35	35	71	4810.36	7	3 - 21	36.0133	434
StgoAgg90	90	195	3025.43	17	4 - 33	36.1489	689
StgoKmn90	90	193	2987.79	18	4 - 33	36.2253	694
StgoSpc90	90	187	3173.74	21	4 - 32	36.3032	712
StgoGau90	90	197	3023.99	17	4 - 33	35.8493	696
StgoAgg135	135	302	2445.21	23	5 - 41	35.9356	814
StgoKmn135	135	300	2449.80	24	5 - 41	35.9875	825
StgoSpc135	135	293	2518.20	26	5 - 40	37.1621	844
StgoGau135	135	308	2429.90	24	5 - 41	35.8734	829

(a) 35 clústeres **(b)** 90 clústeres **(c)** 135 clústeres **(d)** Red de Caminos Aproximada

Figura 5.3: Relajación de la instancia RED al aplicar clustering basado en Mezcla de Gaussianas.

diferencias en áreas periféricas. Estas diferencias parecen reducirse cuando se usa un mayor número de clústeres.

Una de las principales características de nuestro método es que mantiene información de paraderos desde áreas periféricas con baja densidad de paraderos. Estos paraderos deben ser

representados en la instancia incluso si tienen baja demanda. Una selección aleatoria de paraderos como la usada por [15] puede omitir paraderos en estos sectores y dejar ubicaciones sin representación en la instancia.

Un ejemplo de esto puede ser visto en la Figura 5.4 que muestra el área periférica del sector noroeste de Santiago para distintos niveles de relajación y las ubicaciones mezcladas en el escenario original.

(a) 35 clústeres

(b) 90 clústeres

(c) 135 clústeres

(d) Paraderos de RED

Figura 5.4: Comparación de la estructura de la red de caminos sobre un área periférica entre el escenario original de RED y su relajación aplicando Clustering Aglomerativo.

Estos paraderos tienen una probabilidad menor al 1 % de ser escogidos aleatoriamente. El framework mantiene estos paraderos representados para tamaños de instancia medianos (Figura 5.4b) o grandes (Figura 5.4c) aunque para tamaños pequeños (Figura 5.4a) el centroide

podría quedar distante. Con un bajo número de clústeres, los algoritmos tienden a agrupar todas las posibles ubicaciones en un solo clúster produciendo un centroide a una distancia considerable de la mayoría de los paraderos contenidos. Al incrementar el número de clústeres, se permite al algoritmo separar estas ubicaciones en múltiples clústeres con centroides más cercanos a cada una.

Esto también afecta la estructura de caminos de la instancia. Es improbable que se generen enlaces sobre características geográficas o estructuras urbanas, donde no pueden haber caminos (por ejemplo, aeropuertos, colinas, campos agrícolas, etc.), siempre que hayan suficientes clústeres para generar una red de caminos que las evite como ocurre en las Figuras 5.4b y 5.4c. Sin embargo, tal cual se observa en la Figura 5.4a ciertas asignaciones de clustering pueden llevar a un centroide que generará conexiones que no podrán evitar estos obstáculos.

Recordar que los caminos no necesariamente son una aproximación fiel a la realidad pues en la configuración experimental se hizo uso de una red de caminos aproximada generada a partir de la cercanía de los paraderos. No obstante, los resultados mostrados en esta sección muestran que el método para generar la RRN es capaz de mantener una estructura de caminos similar sin importar el origen de los datos.

5.2.3. Distribución de demanda

El análisis de la sección anterior se concentraba en las características de la red de caminos generada y como se mantenían características del escenario original. Otra de las características críticas del procedimiento de relajación es su capacidad de preservar la distribución de demanda para poder generar una buena aproximación al escenario original.

Comparación con datos originales

Se inicia el análisis de distribución de demanda de las instancias reducidas con una comparación de la demanda respecto de los datos originales. Para ello, se presenta la Figura 5.5 que presenta la distribución de la demanda para distintos escenarios.

(a) 35 clústeres **(b)** 90 clústeres **(c)** 135 clústeres **(d)** Zonas RED

Figura 5.5: Comparación de la distribución de demanda entre las instancias generadas usando Clustering Espectral y los datos originales de RED. El color de cada enlace corresponde al flujo de pasajeros viajando a través de la ruta más corta hasta su destino.

Notar que la distribución de demanda sigue un patrón similar respecto al escenario original, captando flujos similares de demanda en el centro de Santiago desde las zonas periféricas en el sur, nororiente y norponiente. El parecido en los flujos de demanda aumenta junto al número de clústeres.

Comparación con instancias existentes

El análisis de la demanda continua con una comparación de las instancias reducidas con aquellas de la literatura, observando los histogramas de demanda. Estos se construyen contando el número de entradas en las matrices de demanda con una cierta cantidad de viajes o usuarios moviéndose entre un par de ubicaciones. Se construye un histograma de demanda análogo para la matriz de demanda zonal simple.

La Figura 5.6 muestra el histograma de demanda para la instancia Mandl. Esta instancia ha sido usada en varios trabajos previos y es la más antigua de la literatura. Esta basada en un caso de estudio de Suiza, sin embargo, no hay mucha información respecto a su construcción o correspondencia con la realidad.

La forma del histograma muestra un fuerte descenso. 107 entradas tienen menos de 18 viajes asociados (47.5 % del número total de entradas en la matriz de demanda de tamaño 15×15)

Figura 5.6: Histograma de demanda de instancia Mandl.

y solo 10 entradas (4.4 %) tienen más de 400 viajes.

La Figura 5.7 muestra los histogramas de demanda para la serie de instancias de Mumford. La matriz de demanda en esta instancia fue generada aleatoriamente tomando valores de una distribución aleatoria uniforme acotada por unos parámetros de usuario [8] [33].

Figura 5.7: Histogramas de matrices de demanda de instancias de Mumford

En el histograma se puede ver que el número de viajes está distribuido de forma más igualitaria entre distintos números de enlaces y no es visible el descenso similar como el que hay en la instancia Mandl y los datos de RED.

En la Figura 5.8a y Figura 5.8b se pueden ver los histogramas de demanda de las instancias Nottingham100 y Edinburgh200. La demanda de estas instancias esta basada en datos

(a) Nottingham100 **(b)** Edinburgh200

Figura 5.8: Histograma de demanda de instancias Nottingham100 y Edinburgh200.

censales que cuentan la cantidad de viajes entre ubicaciones de origen y destino.

Es fácil de observar que la mayoría de los valores de las entradas se encuentran en el rango $[0, 10]$ en ambas instancias y corresponden a valores pares. En el caso de la instancia de Edinburgh200 los valores muestran un descenso de manera similar a como ocurre en la instancia de Mandl, pero la forma es distinta en la de Nottingham en que hay un incremento de la cantidad de pares origen-destino con una mayor cantidad de viajes.

Por último, en la Figura 5.9 aparece una comparación de histogramas de demanda de instancias generadas usando el framework propuesto (aplicando Clustering Aglomerativo) y la matriz de demanda zonal simple (i.e. los datos originales de RED).

Se puede ver que el número de enlaces decrece junto con los niveles de demanda. La tasa de descenso aumenta con un número mayor de clústeres asemejandose en mayor medida a la demanda del caso original.

Al haber menos clústeres, cada centroide cubre más zonas y adquiere una mayor cantidad de demanda. Más pares de clústeres tienen una demanda semejante entre ellos.

Aunque cada incremento en el número de clústeres aumenta la similitud al caso original, hay que reconocer que nunca se igualara. Pensando en el caso extremo donde el número de clústeres se iguala al número de ubicaciones mezcladas, la distribución de demanda no se igualara a los datos originales por la forma en que se penaliza la demanda basado en la distancia entre el centroide del clúster y el centroide de la zona.

Figura 5.9: Histogramas de matrices de demanda de las instancias generadas usando Clustering Aglomerativo con la matriz de demanda zonal simple. La cantidad de elementos esta en escala logarítmica.

Por tanto, el framework es capaz de generar instancias con un patrón de demanda similar al caso original mientras se use un número suficiente de clústeres. La instancia se corresponde más con la realidad respecto de instancias anteriores como las de Mumford. Además las instancias reducidas tienen una distribución de valores más variado que las instancias Nottingham100 y Edinburgh200.

5.2.4. Similitud entre algoritmos de clustering

En la Tabla 5.1 se pudo observar que las métricas de complejidad son bastante similares entre sí para cada escenario generado con cada algoritmo recomendado. Sin embargo, se debe tener en cuenta que igual podría haber diferencias considerables en la RED y que los algoritmos K-Means, Mezcla de Gaussianas y Espectral son estocásticos pues tienen una inicialización aleatoria. Las diferencias en la agrupación entre distintos algoritmos de clustering producen un desplazamiento de los centroides y, por tanto, variaciones en la estructura de la red de caminos reducida y valores en la matriz de demanda. Por lo que a continuación se analiza la similitud de la salida de los algoritmos de clustering.

En la Tabla 5.2 aparece una comparación del ARI entre cada par de algoritmos de clustering. Los valores promedio de esta medida de similitud están todos por encima de 0.5 indicando

que la mayoría de los algoritmos generaron para cada par comparado. El bajo valor de la desviación estándar muestra una baja dispersión de esta medida. Por tanto, las salidas no son idénticas, pero bastante similares, por lo que los clústeres tienen bastantes elementos en común.

Tabla 5.2: ARI entre algoritmos de clustering: promedio y desviación estándar de diez ejecuciones usando 90 clústeres.

ARI Avg	Kmn	Agg	Spc	Gau
Kmn	0.633	0.547	0.554	0.615
Agg	0.547	1.000	0.530	0.546
Spc	0.554	0.530	0.854	0.559
Gau	0.615	0.546	0.559	0.604

ARI SD	Kmn	Agg	Spc	Gau
Kmn	0.022	0.012	0.013	0.025
Agg	0.012	0.000	0.006	0.019
Spc	0.013	0.006	0.023	0.020
Gau	0.025	0.019	0.020	0.032

En la Tabla 5.3 aparece una comparación del UMI entre cada par de algoritmos de clustering. Los valores del UMI son consistentes con aquellos del ARI mostrando alta similitud entre los algoritmos de clustering insinuando pequeñas diferencias estructurales. Más aún, podemos observar que la mayoría (80 % o más) de los centroides tienen una pareja única entre varias salidas a pesar de sus diferencias.

Tabla 5.3: UMI entre algoritmos de clustering: promedio y desviación estándar de diez ejecuciones usando 90 clústeres.

UMI Avg	Kmn	Agg	Spc	Gau
Kmn	0.823	0.811	0.766	0.817
Agg	0.822	1.000	0.783	0.816
Spc	0.764	0.746	0.924	0.767
Gau	0.817	0.817	0.770	0.801

UMI SD	Kmn	Agg	Spc	Gau
Kmn	0.026	0.033	0.029	0.037
Agg	0.026	0.000	0.028	0.032
Spc	0.026	0.028	0.024	0.034
Gau	0.032	0.023	0.037	0.039

Capítulo 6

Conclusiones y Perspectivas

En este trabajo, se aborda la generación de instancias para el UTRP usando datos de demanda de un escenario real analizando los problemas de instancias bien conocidas. Desde que algunas de estas instancias tienen pocas similitudes con sus contrapartes de la vida real, pueden ser denotadas como aproximaciones de baja calidad. Otras instancias ocupaban datos reales, pero la reducción era insuficiente.

En este trabajo se propone un framework para la creación de instancias para el UTRP que simplifica las instancias creadas mientras mantiene características críticas como la estructura de red de caminos y la distribución de demanda de los pasajeros. Este framework asume que la información de demanda es conocida o estimada, y que las ubicaciones geográficas de los paraderos o de los sitios de interés fueron previamente establecidas. No obstante, en la aplicación del framework no se usó información de terrenos o de la red de caminos, en su lugar usándose una aproximación de la red de caminos obtenida por medio de K-neighbors.

Al realizar una inspección visual de las instancias generadas con el framework, se observó que estas incluían ubicaciones periféricas en la RRN, incluso si es que había pocos paraderos asociados. Así la instancia es capaz de representar mejor la estructura de la ciudad que acercamientos previos. La evidencia empírica también muestra que las instancias generadas mantienen un comportamiento de demanda similar respecto a los datos de demanda reales.

El framework está basado en el uso de un algoritmo de clustering. Variados algoritmos fueron

probados, cuatro de los cuales fueron recomendados pues permitían obtener una relajación adecuada con un nivel de dificultad conocido al permitir ingresar el número de clústeres. Los algoritmos que fueron descartados no se ajustaban bien al caso de estudio: descartaban ubicaciones periféricas como ruido, no eran capaces de clasificar datos homogéneos y/o necesitaban un cuidadoso ajuste de hiper-parámetros. Los algoritmos recomendados presentaban una salida y rendimiento similares, aun cuando eran bastante distintos entre sí.

6.1. Perspectivas

Desde este punto en adelante múltiples vías de investigación son posibles: escalar la solución de la instancia reducida al escenario original, revisitar la construcción de la RRN con sistema de georreferenciación para una mejor aproximación, y probar algoritmos descartados por requerir ajuste de parámetros.

Es de vital importancia, la integración del framework en técnicas de resolución para abordar escenarios de escala real como es el caso de RED. Para ello es necesario, una vez obtenida una solución sobre la instancia reducida, aprovechar su información escalándola hacia el caso real más complejo y así apoyar la generación de una solución útil en el diseño de un sistema de transporte urbano. Una posibilidad para esto es aprovechar un subproducto del Clustering Aglomerativo que es la jerarquía de clústeres, que indica las mezclas sucesivas clústeres de los nodos hasta la obtención de los clústeres finales.

Otra posibilidad es refinar las instancias existentes usando información de caminos y terrenos desde sistemas de información geográfica para obtener una mejor aproximación. Esto es requerido si se quieren evitar enlaces que crucen sobre características geográficas pequeñas como ríos y canales, no detectables al usar la aproximación de caminos usada en el presente trabajo. Adicionalmente, se podrían asignar distancias más precisas entre los nodos que incluso tengan en cuenta otros factores como la capacidad de tráfico y diferencias en velocidad vehicular.

Por último, también se podría reestudiar el comportamiento de los algoritmos de clustering

como Mean Shift y Affinity Propagation que requerían un procedimiento de ajuste de hiper-parámetros.

Apéndice A

Resultados completos

En este apéndice se muestran aquellos resultados y figuras que, para facilitar la lectura, fueron omitidos de la presentación de resultados en la Sección 5.2.

A.1. Redes de Caminos Relajadas

Para comenzar se muestran la Red Aproximada de Caminos y las RRN de todas las instancias presentadas en la Tabla 5.1. En cada figura se muestran los bordes de las comunas y zonas urbanas de la ciudad de Santiago [53].

Figura A.1: Red Aproximada de Caminos a partir de los datos de RED.

(a) 35 clústeres **(b)** 90 clústeres **(c)** 135 clústeres

Figura A.2: RRN de instancias generadas con K-Means

(a) 35 clústeres **(b)** 90 clústeres **(c)** 135 clústeres

Figura A.3: RRN de instancias generadas con Clustering Aglomerativo

(a) 35 clústeres **(b)** 90 clústeres **(c)** 135 clústeres

Figura A.4: RRN de instancias generadas con Clustering basado en mezcla de Gaussianas

(a) 35 clústeres **(b)** 90 clústeres **(c)** 135 clústeres

Figura A.5: RRN de instancias generadas con Clustering Espectral

A.2. Comportamiento de la Demanda

En esta sección se muestran los mapas de calor donde se visualiza el comportamiento de la demanda, tanto entre las zonas de RED como de todas las instancias generadas presentadas en la Tabla 5.1. En cada figura se muestran los bordes de las comunas y zonas urbanas de la ciudad de Santiago [53].

Figura A.6: Distribución de demanda de las zonas de RED

(a) 35 clústeres **(b)** 90 clústeres **(c)** 135 clústeres

Figura A.7: Comportamiento de demanda al usar K-Means

(a) 35 clústeres **(b)** 90 clústeres **(c)** 135 clústeres

Figura A.8: Comportamiento de demanda al usar Clustering Aglomerativo

(a) 35 clústeres **(b)** 90 clústeres **(c)** 135 clústeres

Figura A.9: Comportamiento de demanda al usar Clustering basado en mezcla de Gaussianas

(a) 35 clústeres **(b)** 90 clústeres **(c)** 135 clústeres

Figura A.10: Comportamiento de demanda al usar Clustering Espectral

A.3. Histogramas de demanda

En lo que sigue se muestran los histogramas de demanda donde se visualiza el comportamiento de la demanda de todas las instancias generadas presentadas en la Tabla 5.1.

Figura A.11: Histogramas de matrices de demanda de las instancias generadas usando K-Means con la matriz de demanda zonal simple. La cantidad de elementos esta en escala logarítmica.

Figura A.12: Histogramas de matrices de demanda de las instancias generadas usando Clustering Aglomerativo con la matriz de demanda zonal simple. La cantidad de elementos esta en escala logarítmica.

Figura A.13: Histogramas de matrices de demanda de las instancias generadas usando Clustering basado en Mezcla de Gaussianas con la matriz de demanda zonal simple. La cantidad de elementos esta en escala logarítmica.

Figura A.14: Histogramas de matrices de demanda de las instancias generadas usando Clustering Espectral con la matriz de demanda zonal simple. La cantidad de elementos esta en escala logarítmica.

A.4. Similitud entre algoritmos de clustering

En esta sección se muestran las tablas con los valores obtenidos para ARI y UMI para todos los valores probados de números clústeres: 35, 90 y 135.

Tabla A.1: ARI entre algoritmos de clustering: promedio y desviación estándar de diez ejecuciones usando 35 clústeres.

ARI Avg	Kmn	Agg	Spc	Gau
Kmn	0.673	0.568	0.509	0.625
Agg	0.568	1.000	0.482	0.542
Spc	0.509	0.482	0.911	0.498
Gau	0.625	0.542	0.498	0.593

ARI SD	Kmn	Agg	Spc	Gau
Kmn	0.041	0.016	0.025	0.056
Agg	0.016	0.000	0.011	0.021
Spc	0.025	0.011	0.056	0.028
Gau	0.056	0.021	0.028	0.054

Tabla A.2: ARI entre algoritmos de clustering: promedio y desviación estándar de diez ejecuciones usando 90 clústeres.

ARI Avg	Kmn	Agg	Spc	Gau
Kmn	0.633	0.547	0.554	0.615
Agg	0.547	1.000	0.530	0.546
Spc	0.554	0.530	0.854	0.559
Gau	0.615	0.546	0.559	0.604

ARI SD	Kmn	Agg	Spc	Gau
Kmn	0.022	0.012	0.013	0.025
Agg	0.012	0.000	0.006	0.019
Spc	0.013	0.006	0.023	0.020
Gau	0.025	0.019	0.020	0.032

Tabla A.3: ARI entre algoritmos de clustering: promedio y desviación estándar de diez ejecuciones usando 135 clústeres.

ARI Avg	Kmn	Agg	Spc	Gau
Kmn	0.605	0.559	0.569	0.591
Agg	0.559	1.000	0.553	0.551
Spc	0.569	0.553	0.785	0.554
Gau	0.591	0.551	0.554	0.581

ARI SD	Kmn	Agg	Spc	Gau
Kmn	0.023	0.008	0.018	0.027
Agg	0.008	0.000	0.007	0.019
Spc	0.018	0.007	0.019	0.015
Gau	0.027	0.019	0.015	0.024

Tabla A.4: UMI entre algoritmos de clustering: promedio y desviación estándar de diez ejecuciones usando 35 clústeres.

UMI Avg	Kmn	Agg	Spc	Gau
Kmn	0.853	0.834	0.676	0.797
Agg	0.817	1.000	0.611	0.754
Spc	0.659	0.623	0.961	0.653
Gau	0.796	0.777	0.653	0.746

UMI SD	Kmn	Agg	Spc	Gau
Kmn	0.048	0.031	0.044	0.069
Agg	0.043	0.000	0.023	0.085
Spc	0.047	0.011	0.031	0.057
Gau	0.059	0.060	0.072	0.079

Tabla A.5: UMI entre algoritmos de clustering: promedio y desviación estándar de diez ejecuciones usando 90 clústeres.

UMI Avg	Kmn	Agg	Spc	Gau
Kmn	0.823	0.811	0.766	0.817
Agg	0.822	1.000	0.783	0.816
Spc	0.764	0.746	0.924	0.767
Gau	0.817	0.817	0.770	0.801

UMI SD	Kmn	Agg	Spc	Gau
Kmn	0.026	0.033	0.029	0.037
Agg	0.026	0.000	0.028	0.032
Spc	0.026	0.028	0.024	0.034
Gau	0.032	0.023	0.037	0.039

Tabla A.6: UMI entre algoritmos de clustering: promedio y desviación estándar de diez ejecuciones usando 135 clústeres.

UMI Avg	Kmn	Agg	Spc	Gau
Kmn	0.819	0.805	0.782	0.801
Agg	0.784	1.000	0.740	0.771
Spc	0.772	0.737	0.896	0.752
Gau	0.801	0.767	0.763	0.780

UMI SD	Kmn	Agg	Spc	Gau
Kmn	0.026	0.021	0.023	0.031
Agg	0.031	0.000	0.021	0.020
Spc	0.023	0.015	0.020	0.025
Gau	0.030	0.028	0.026	0.031

Bibliografía

[1] M.-C. Shih and H. S. Mahmassani, "A design methodology for bus transit networks with coordinated operations," tech. rep., 1994.

[2] P. N. Kechagiopoulos and G. N. Beligiannis, "Solving the urban transit routing problem using a particle swarm optimization based algorithm," in *Applied Soft Computing 21*, pp. 654–676, 2014.

[3] R. J. Lee and I. N. Sener, "Transportation planning and quality of life: Where do they intersect?," *Transport policy*, vol. 48, pp. 146–155, 2016.

[4] P. Chakroborty, "Genetic algorithms for optimal urban transit network design," *Computer-Aided Civil and Infrastructure Engineering*, vol. 18, no. 3, pp. 184–200, 2003.

[5] V. Guihaire and J.-K. Hao, "Transit network design and scheduling: A global review," *Transportation Research Part A: Policy and Practice*, vol. 42, no. 10, pp. 1251–1273, 2008.

[6] A. Ceder and N. H. Wilson, "Bus network design," *Transportation Research Part B: Methodological*, vol. 20, no. 4, pp. 331–344, 1986.

[7] T. L. Magnanti and R. T. Wong, "Network design and transportation planning: Models and algorithms," *Transportation science*, vol. 18, no. 1, pp. 1–55, 1984.

[8] C. L. Mumford, "New heuristic and evolutionary operators for the multi-objective urban transit routing problem," in *2013 IEEE Congress on Evolutionary Computation*, pp. 939–946, 2013.

[9] P. Chakroborty and T. Wivedi, "Optimal route network design for transit systems using genetic algorithms," *Engineering optimization*, vol. 34, no. 1, pp. 83–100, 2002.

[10] L. Fan and C. L. Mumford, "A metaheuristic approach to the urban transit routing problem," *Journal of Heuristics*, vol. 1, no. 16, pp. 353–372, 2010.

[11] L. Fan, C. L. Mumford, and D. Evans, "A simple multi-objective optimization algorithm for the urban routing transit problem," in *IEEE Congress on Evolutionary Computation*, School of Computer Science, Cardiff University, UK, 2009.

[12] M. P. John, C. L. Mumford, and R. Lewis, "An improved multi-objective algorithm for the urban transit routing problem," in *European Conference on Evolutionary Computation in Combinatorial Optimization*, pp. 49–60, Springer, 2014.

[13] I. M. Cooper, M. P. John, R. Lewis, C. L. Mumford, and A. Olden, "Optimising large scale public transport network design problems using mixed-mode parallel multi-objective evolutionary algorithms," in *2014 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2841–2848, IEEE, 2014.

[14] L. Ahmed, C. Mumford, and A. Kheiri, "Solving urban transit route design problem using selection hyper-heuristics," *European Journal of Operational Research*, vol. 274, no. 2, pp. 545–559, 2019.

[15] M. P. John, *Metaheuristics for designing efficient routes & schedules for urban transportation networks.* PhD thesis, Cardiff University, 2016.

[16] P. H. Soares, C. L. Mumford, K. Amponsah, and Y. Mao, "An adaptive scaled network for public transport route optimisation," *Public Transport*, vol. 11, no. 2, pp. 379–412, 2019.

[17] J. C. Muñoz, M. Batarce, and D. Hidalgo, "Transantiago, five years after its launch," *Research in Transportation Economics*, vol. 48, pp. 184–193, 2014.

[18] Red Metropolitana de Movilidad, "Información del Sistema | Red Metropolitana de Movilidad." http://www.red.cl/acerca-de-red/informacion-del-sistema, 2020.

[19] A. E. Eiben, J. E. Smith, *et al.*, *Introduction to evolutionary computing*, vol. 53. Springer, 2003.

[20] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM computing surveys (CSUR)*, vol. 35, no. 3, pp. 268–308, 2003.

[21] M. S. Levin, "Combinatorial optimization in system configuration design," *Automation and Remote Control*, vol. 70, no. 3, p. 519, 2009.

[22] C. A. C. Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," *Computer methods in applied mechanics and engineering*, vol. 191, no. 11-12, pp. 1245–1287, 2002.

[23] C. A. C. Coello, G. B. Lamont, D. A. Van Veldhuizen, *et al.*, *Evolutionary algorithms for solving multi-objective problems*, vol. 5. Springer, 2007.

[24] M. Caramia and P. Dell'Olmo, "Coloring graphs by iterated local search traversing feasible and infeasible solutions," *Discrete Applied Mathematics*, vol. 156, no. 2, pp. 201–217, 2008.

[25] A. Blot, M.-É. Kessaci, and L. Jourdan, "Survey and unification of local search techniques in metaheuristics for multi-objective combinatorial optimisation," *Journal of Heuristics*, vol. 24, no. 6, pp. 853–877, 2018.

[26] M. Emmerich, N. Beume, and B. Naujoks, "An emo algorithm using the hypervolume measure as selection criterion," in *International Conference on Evolutionary Multi-Criterion Optimization*, pp. 62–76, Springer, 2005.

[27] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms—a comparative case study," in *International conference on parallel problem solving from nature*, pp. 292–301, Springer, 1998.

[28] O. J. Ibarra-Rojas, F. Delgado, R. Giesen, and J. C. Muñoz, "Planning, operation, and control of bus transport systems: A literature review," *Transportation Research Part B: Methodological*, vol. 77, pp. 38–75, 2015.

[29] I. Contreras and E. Fernández, "General network design: A unified view of combined location and network design problems," *European Journal of Operational Research*, vol. 219, no. 3, pp. 680–697, 2012.

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

[31] R. Z. Farahani, E. Miandoabchi, W. Y. Szeto, and H. Rashidi, "A review of urban transportation network design problems," *European Journal of Operational Research*, vol. 229, no. 2, pp. 281–302, 2013.

[32] C. E. Mandl, "Evaluation and optimization of urban public transportation networks," *European Journal of Operational Research*, vol. 5, no. 6, pp. 396–404, 1980.

[33] C. Mumford, "Supplementary material for: New heuristic and evolutionary operators for the multi-objective urban transit routing problem, cec 2013," *Cancun, Mexico: IEEE*, 2016.

[34] C. Mandl, "Applied network optimization," 1980.

[35] C. D. Manning, H. Schütze, and P. Raghavan, *Introduction to information retrieval*. Cambridge university press, 2008.

[36] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," tech. rep., Stanford, 2006.

[37] H. Joe and J. Ward, "Hierarchical grouping to optimize an objective function," *J Am Stat Assoc*, vol. 58, no. 301, pp. 236–244, 1963.

[38] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[39] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[40] J. A. Bilmes *et al.*, "A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models," *International Computer Science Institute*, vol. 4, no. 510, p. 126, 1998.

[41] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[42] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *science*, vol. 315, no. 5814, pp. 972–976, 2007.

[43] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," *ACM Sigmod record*, vol. 28, no. 2, pp. 49–60, 1999.

[44] E. Schubert and M. Gertz, "Improving the cluster structure extracted from optics plots.," in *LWDA*, pp. 318–329, 2018.

[45] D. Steinley, "Properties of the hubert-arable adjusted rand index.," *Psychological methods*, vol. 9, no. 3, p. 386, 2004.

[46] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical association*, vol. 66, no. 336, pp. 846–850, 1971.

[47] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *The Journal of Machine Learning Research*, vol. 11, pp. 2837–2854, 2010.

[48] L. Hubert and P. Arabie, "Comparing partitions," *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.

[49] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[50] Directorio de Transporte Público Metropolitano, "Matrices de viaje." http://www.dtpm.gob.cl/index.php/documentos/matrices-de-viaje, 2017.

[51] Directorio de Transporte Público Metropolitano, "Informe de gestión 2017." http://www.dtpm.gob.cl/index.php/documentos/informes-de-gestion, 2017. Last Access: June 22, 2020.

[52] P. Correa, "Un día en el recorrido más largo del transantiago." https://www.t13.cl/noticia/nacional/un-dia-en-el-recorrido-mas-largo-del-transantiago, 2017. Last Access: June 22, 2020.

[53] C. Albers, "Coberturas sig, chile continental." `http://labgeo.ufro.cl/fichas/chile_geo/ficha_cl_geo.html`, 2019. Last Access: July 07, 2020.