

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



“PROPUESTA ADAPTIVA DE UN ALGORITMO TOPOLÓGICO PARA LA RESOLUCIÓN DE AMBIGÜEDAD ESPACIAL”

ROBERTO COVARRUBIAS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Roberto León
Profesor Correferente: Elizabeth Montero

Agosto - 2024

DEDICATORIA

A mi familia, compañeros y amigos que me han apoyado durante toda mi trayecto en la universidad.

AGRADECIMIENTOS

A mi familia, mis padres, mis hermanas y sobrinas, por su apoyo y compañía durante toda mi vida y mi estadía en la universidad.

A mi profesor guía, Roberto León, y mi profesora correferente, Elizabeth Montero, por darme la oportunidad de tener este trabajo de investigación y por darme su guía en los nuevos temas y desafíos que conllevo esta memoria.

A mi primo Roberto Ordenes y a mi amigo Cesar Araya, los cuales me ayudaron en los primeros días de mi vida como universitario independiente.

Y a mis compañeros de universidad que fui conociendo a lo largo de estos años, a mis compañeros de PACE, compañeros y amigos de la carrera Ingeniería Civil Informática, los cuales pude llevar mi estadía y experiencias dentro de la universidad convirtiéndose en amigos personales.

RESUMEN

Resumen— Los algoritmos de resolución de ambigüedad espacial, como los *Topological Map-Matching Algorithms*, son clave para abordar el problema de asignación de calles que poseen los Sistemas de Transporte Inteligente que buscan mejorar la eficiencia y seguridad del transporte público, recolección de residuos y sistemas de peaje electrónico. No obstante, estos algoritmos cuentan con la dificultad de configurar sus parámetros necesarios para su correcta ejecución.

En este trabajo de memoria se propone e implementa un algoritmo adaptativo para la resolución de esta problemática con la capacidad de configurar los parámetros necesarios frente a las diferentes situaciones de datos GPS.

Palabras Clave— Ambigüedad espacial, Map-matching, TMMA, *accuracy*, *buffer*, *tol*, Algoritmo Adaptativo, Sintonización de Parámetros, Modelos de Regresión.

ABSTRACT

Abstract— Spatial ambiguity resolution algorithms, such as Topological Map-Matching Algorithms, are crucial for addressing the street assignment problem faced by Intelligent Transportation Systems aimed at improving the efficiency and safety of public transport, waste collection, and electronic toll systems. However, these algorithms face the challenge of configuring their parameters correctly for effective execution.

This work proposes and implements an adaptive algorithm to address this issue, capable of configuring necessary parameters in response to various GPS data scenarios.

Keywords— Spatial ambiguity, Map-matching, TMMA, *Accuracy*, *buffer*, *tol*, Adaptive Algorithm, Parameter Tuning, Regression Models.

ÍNDICE DE CONTENIDOS

RESUMEN	IV
ABSTRACT	IV
ÍNDICE DE FIGURAS	VII
ÍNDICE DE TABLAS	VII
INTRODUCCIÓN	1
CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA	2
1.1 Objetivos	4
1.1.1 Objetivos Específicos	4
CAPÍTULO 2: MARCO CONCEPTUAL	5
2.1 Problema de Asignación de Calles	5
2.2 Map-Matching Algorithms	6
2.3 Topological Map-Matching Algorithms	8
2.4 Importancia de los parámetros en los MMAs	9
2.5 Algoritmos de sintonización	11
2.5.1 Sintonización de parámetros	11
2.5.2 Estado del arte	12
2.5.3 Evoca	13
2.5.4 ParamILS	15
2.5.5 Irace	17
2.6 Resumen del Capitulo	20
CAPÍTULO 3: PROPUESTA DE SOLUCIÓN	21
3.1 Proceso de trabajo	21
3.1.1 ArcGIS	23
3.1.2 Parámetros a ajustar	23
3.2 Fundamentos del uso e implementación de algoritmos sintonizadores	24
3.2.1 Evoca	24
3.2.2 ParamILS	25
3.2.3 Irace	25
3.2.4 Metodología de Experimentación	26
3.3 Implementación de la solución	26
3.3.1 Modelos de Regresión	26
3.4 Resumen	28
CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN	29
4.1 Experimentos de Algoritmos Sintonizadores	29
4.1.1 Instancias de Datos	30

4.1.2	Entorno de Ejecución	30
4.2	Resultados de Sintonización	31
4.2.1	Mejores Soluciones	32
4.2.2	Otras Configuraciones de Parámetros	33
4.3	Modelos de Regresión Lineal	34
4.3.1	Instancias de Datos	37
4.3.2	Modelos de Regresión Lineal	37
4.3.3	Entorno de Ejecución	37
4.3.4	Metodología de Experimentación	38
4.4	Resultados de Experimentos con Modelos de Regresión	39
4.4.1	Resultados Según Función	40
4.4.2	Resultados Según Sintonizador	42
4.4.3	Resultados Modelos ParamILS	43
4.5	Resultados Parámetros Genéricos y Modelo adaptativo	45
4.6	Conclusiones del Capítulo	47
CAPÍTULO 5: CONCLUSIONES		48
5.1	Conclusiones Generales	48
5.2	Cumplimiento de objetivos	48
5.2.1	Objetivos Específicos	49
5.3	Trabajo Futuro	50
ANEXOS		51
REFERENCIAS BIBLIOGRÁFICAS		59

ÍNDICE DE FIGURAS

1	Imagen de una calle que es dividida en 2. Fuente: “ <i>Pre-Processing Techniques for Improving a Topological Map-Matching Algorithm</i> ” de Blazquez, León, Pérez y Delgado [Blazquez et al., 2022].	6
2	Diagrama de flujo de TMMA. Fuente: “ <i>An Instance-Specific Parameter Tuning Approach Using Fuzzy Logic for a Post-Processing Topological Map-Matching Algorithm</i> ” Blazques, C., Ries, J., Miranda, P.A., León, R. J. [Blazquez et al., 2018].	9
3	Diagrama de integración de Evoca. Fuente: “ <i>A New Algorithm for Reducing Metaheuristic Design Effort</i> ” María-Cristina Riff and Elizabeth Montero [Riff y Montero, 2013].	16
4	Diagrama de flujo de trabajo. Fuente: Elaboración propia	22
5	Diagrama del proceso de generación de modelos de regresión usando la data de los procesos de sintonización. Fuente: “Elaboración propia”	27
6	Mapa de Los Puntos GPS en el Mapa de ArcMap. Fuente: “Extracto de ArcMap”	31
7	Gráfica exactitud vs Conjunto de Datos. Fuente: Elaboración propia	32
8	Diagrama de Modelos con el Cálculo de Componentes según sus Funciones. Fuente: Elaboración propia	35
9	Mapa de Los Puntos GPS en el Mapa de ArcMap. Fuente: “Extracto de ArcMap”	38
10	Diagrama Representativo de Experimentos de Modelos de Regresión Lineal. Fuente: Elaboración propia.	39
11	Gráfico de Dispersión de los Modelos <i>Min</i> para Evoca, ParamILS e Irace. Fuente: Elaboración propia	40
12	Gráfico de Resultados de TMMA Adaptativo con Modelos ParamILS/ <i>Mean</i> y TMMA con valores genéricos de la literatura. Fuente: Elaboración propia . .	46
13	Gráfico de Dispersión de los Modelos <i>Max</i> para Evoca, ParamILS e Irace. Fuente: Elaboración propia	52
14	Gráfico de Dispersión de los Modelos <i>Mean</i> para Evoca, ParamILS e Irace. Fuente: Elaboración propia	53
15	Gráfico de Dispersión de los Modelos <i>STD</i> para Evoca, ParamILS e Irace. Fuente: Elaboración propia	53
16	Gráfico de Dispersión de los Modelos <i>Max_Max_STD_STD</i> para Evoca, ParamILS e Irace. Fuente: Elaboración propia	54
17	Gráfico de Dispersión de los Modelos <i>Max_STD_Max_Min</i> para Evoca, ParamILS e Irace. Fuente: Elaboración propia	54
18	Gráfico de Dispersión de los Modelos <i>Min_Mean_Min_Mean</i> para Evoca, ParamILS e Irace. Fuente: Elaboración propia	55
19	Gráfico de Dispersión de los Modelos <i>Mean_Min_STD_STD</i> para Evoca, ParamILS e Irace. Fuente: Elaboración propia	55

ÍNDICE DE TABLAS

1	Instancias de Datos Usados en la Sintonización y Entrenamiento de Modelos de Regresión.	30
2	Resultados de los Sintonizadores.	32
3	Segundas y Terceras Mejores Configuraciones.	34
4	Estadísticas de Modelos según Funciones	41
5	Estadísticas de Modelos según Funciones	41
6	Estadísticas de Modelos según Sintonizador.	43
7	Estadísticas de Modelos de ParamILS según Funciones	44
8	Estadísticas de Modelos de ParamILS según Funciones	44
9	Diferencias de Resultados del TMMA Adaptativo con Modelos ParamILS/ <i>Mean</i> y TMMA con valores genéricos.	46
10	Conjuntos de Instancias 1, 2 y 3 de Datos de Prueba.	51
11	Conjuntos de Instancias 4 y 5 de Datos de Prueba.	51
12	Coeficientes e Intercepto de Modelos de Predicción del Parámetro <i>tol</i>	56
13	Coeficientes e Intercepto de Modelos de Predicción del Parámetro <i>buffer</i> . .	57
14	Resultados de TMMA Adaptativo con Modelos ParamILS/ <i>Mean</i> y TMMA Genérico.	58

INTRODUCCIÓN

Los Sistemas de Transporte Inteligente (ITS) mejoran la eficiencia y seguridad del transporte público, recolección de residuos y sistemas de peaje electrónico mediante tecnologías que integran información y comunicación. Utilizan mediciones GPS integradas en vehículos dentro de un Sistema de Información Geográfica (SIG) para gestionar rutas, aunque enfrentan el desafío del *Map-Matching Problem*, causado por errores en mediciones GPS y la complejidad de la red vial.

Los algoritmos de resolución de ambigüedad espacial, como los *Topological Map-Matching Algorithms* (**TMMAs**), son clave para abordar este problema utilizando los datos GPS. Sin embargo, enfrentan desafíos en la configuración de parámetros como el tamaño del *buffer*, que define el radio de selección de calles candidatas, y la tolerancia de velocidad *tol*, que descarta calles según la velocidad registrada.

En este trabajo de memoria se diseña, implementa y evalúa un algoritmo adaptativo del **TMMA** con la capacidad de configurar los parámetros frente a las diferentes mediciones GPS a las que se enfrente.

Se presenta la definición del problema a abordar y cuál es la necesidad que lo origina, indicando el contexto y los objetivos de este trabajo.

A continuación se presenta el marco conceptual, el cual abarca tanto el estado del arte de los problemas de resolución de ambigüedad espacial, así como los algoritmos sintonizadores de parámetros para indicar la conexión entre ellos.

Posteriormente, se presenta la propuesta de solución, donde se muestra la metodología a seguir para cumplir los objetivos. Se presenta una serie de etapas a seguir para conseguir buenas configuraciones objetivo, la capacidad de adaptación y posterior validación de este proceso.

Tras lo anterior, se realizan las pruebas necesarias para generar la solución planteada, exhibiendo los resultados de las diferentes etapas y comparando el rendimiento del algoritmo adaptativo frente a una configuración previamente estudiada en la literatura de los **TMMA**.

Finalmente, se exponen las conclusiones, las cuales se centran en el valor entregado por la solución, desde la comparación realizada con anterioridad, el cumplimiento de los objetivos planteados y propuestas de trabajo a futuro a consecuencia de lo obtenido en este trabajo.

CAPÍTULO 1

DEFINICIÓN DEL PROBLEMA

Algunos servicios de transporte, como el transporte público, la recolección de residuos, sistemas de peaje electrónico, entre otros, cuentan con Sistemas de Transporte Inteligente (ITS), tecnologías que buscan mejorar la eficiencia, seguridad y sostenibilidad del transporte mediante la información y comunicación. Los ITS utilizan mediciones de Sistema de Posicionamiento Global, GPS (*Global Positioning System*), capturadas por los vehículos, sistema integrado en un entorno de Sistema de Información Geográfica (SIG), las cuales permiten determinar las posiciones de los vehículos. Sin embargo, las mediciones GPS indican la posición del vehículo, las cuales no siempre son asociadas a la vía correcta. Esto se debe a errores en las mediciones GPS y a la complejidad en la red de carreteras, a este problema se le conoce como *Map-Matching Problem* o problema de asignación de calles.

Para resolver este tipo de problemas se utilizan algoritmos de resolución de ambigüedad espacial como los *Topological Map-Matching Algorithms (TMMAs)* [Blazquez *et al.*, 2022]. Los **TMMAs** trabajan con las mediciones GPS, incluyendo la velocidad y posición del vehículo y con un mapa geográfico del lugar recorrido del vehículo. Estas mediciones son entregadas al algoritmo, donde la salida se espera la asignación de la posición del vehículo a alguna vía. El problema radica que estos algoritmos manejan parámetros como el *buffer size*, que considera el radio de selección de calles candidatas para la asignación, y la *tolerancia de velocidad*, que verifica que la velocidad promedio entre dos posiciones sea cercana a la velocidad de la ruta de las posiciones asignadas a las calles. Según esta tolerancia, el algoritmo descarta calles candidatas, así como también acepta las asignaciones calculadas.

La tarea de búsqueda de estos parámetros es esencial para el correcto funcionamiento de los algoritmos y obtención de buenos resultados, sin embargo, no es una tarea sencilla encontrar los parámetros óptimos, pero lo que realmente es un problema es que estos parámetros no pueden ser generalizados, es decir, el algoritmo puede entregar malos resultados aún con parámetros óptimos debido a las diferentes situaciones de donde se obtienen los datos. Por ejemplo, los datos entregados por un taxi que recorra una zona urbana, en la cual existe un alto número de cuadras, semáforos que lo detengan y el límite de velocidad van a enfrentar un problema diferente a mediciones de un camión que recorre una carretera en la cual se produce unas ramificaciones de múltiples calles en distintas direcciones.

Encontrar los parámetros óptimos ante las diferentes situaciones, es el desafío que se busca abordar. En este trabajo se propone generar una versión adaptativa del algoritmo de resolución de ambigüedad espacial con la capacidad de aprender de las características de los datos y adaptar los parámetros según el problema a resolver.

Para abordar el problema anterior, una alternativa es utilizar *algoritmos sintonizadores*, también conocidos como *algoritmos de ajustes de parámetros*, técnicas utilizadas en el aprendizaje automático a fin de conseguir un algoritmo capaz de adaptarse a las diferentes situa-

ciones. Algunos de los métodos considerados en este trabajo son Evoca, ParamILS e Irace.

Evoca [Riff y Montero, 2013] es una herramienta sintonizadora de parámetros que permite a diseñadores de algoritmos no experimentados la capacidad de obtener buenos resultados sin tener un alto conocimiento en métodos de sintonización. Se basa en un algoritmo evolutivo de población, que durante la búsqueda, se seleccionan dos valores de cruce atractivos para su evolución, a fin de obtener parámetros que produzcan buenos resultados. *ParamILS* [Hutter *et al.*, 2007] corresponde a un algoritmo de búsqueda local iterativa que se mueve en un espacio de configuraciones discreto, el cual converge a una configuración de parámetros óptima global pudiendo ser aplicado a diversos tipos de problemas entregando buenos resultados, demostrando una versatilidad en la variedad de problemas a resolver. Finalmente, *Irace* [López-Ibáñez *et al.*, 2016a], un algoritmo sintonizador que es una extensión de los métodos F-Race, se basa en carreras iterativas. A partir de una distribución determinada, *Irace* genera un muestreo de nuevas configuraciones, que son evaluadas en carreras. Luego, la distribución de muestreo se actualiza, sesgándola hacia las mejores configuraciones, buscando así priorizar un espacio de búsqueda más prometedor.

Las técnicas anteriormente mencionadas son vitales para el desarrollo de este trabajo, buscando su análisis, comprensión e implementación en el problema de ajustar parámetros para los **TMMAs**, a fin de poder conseguir resultados óptimos del algoritmo adaptativo que este trabajo busca proponer.

El estudio y el entendimiento de algoritmos para aplicarlos a un contexto de carreteras siempre me fue altamente interesante, ahora los resultados que se puedan obtener pueden contribuir a este campo.

1.1. Objetivos

Diseñar, implementar y evaluar el rendimiento de un algoritmo adaptativo para abordar el problema de resolución de ambigüedad espacial para diferentes rutas.

1.1.1. Objetivos Específicos

- Diseñar un algoritmo para la selección de parámetros del **TMMAs** que se adapte al tipo de ruta.
- Implementar el algoritmo propuesto integrando el TMMA y algoritmos sintonizadores.
- Evaluar el rendimiento del algoritmo propuesto para al menos tres rutas.

CAPÍTULO 2

MARCO CONCEPTUAL

En el presente capítulo se presenta la literatura existente con respecto al problema de asignación de calles. Se presenta también los algoritmos que se utilizan para abordar este problema, tales como los *Map-Matching Algorithms* (MMA) y particularmente los *Topological Map-Matching Algorithms* (TMMA). Este trabajo está enfocado en el problema de ajustar los parámetros utilizados en estos algoritmos, por lo mismo, se presenta también una descripción formal de la sintonización de parámetros, algoritmos y técnicas utilizadas en ellos, enfocándose principalmente en *Evoca*, *ParamILS* e *Irace*.

2.1. Problema de Asignación de Calles

Los servicios ITS buscan marcar rutas por las cuales han transitados los vehículos afiliados a estos sistemas a fin de entregar una mejora en la gestión del tráfico, información al viajero o seguridad y eficiencia a servicios públicos. Para este fin se hace uso del GPS para obtener las posiciones que el vehículo ha recorrido.

El uso del GPS en servicios de ITS es esencial para su funcionamiento en el transporte, no obstante, es una herramienta que para la medición de coordenadas puede verse afectada por interferencias, dada su naturaleza inalámbrica, entregando así un resultado impreciso que puede acarrear múltiples errores al momento de generar una ruta. Se puede creer que este problema se solucionaría con la simpleza de asignar la calle más cercana a la medición GPS que posee el error, pero esto fallaría en casos con una complejidad ligeramente más alta que seguir una carretera recta, es decir, en un alto número de casos.

En la Figura 1 se pueden observar 3 calles y 7 puntos circulares, de los cuales 3 corresponden a mediciones GPS, las cuales están representadas por círculos de color verde con la simbología p_i , donde la numeración es el orden que poseen las mediciones. La calle posteriormente se divide en dos bifurcaciones, lo cual dificulta determinar correctamente el camino recorrido por el vehículo si se utiliza un criterio de asignar la calle más cercana a la medición. La asignación de calle se representa con un círculo marrón con la simbología s_i por sobre las líneas que referencia las calles. Como se puede observar, se determinan los puntos s_1 , s_2 y s_3 como el camino recorrido, pero s_2 está en una calle distinta a s_1 lo cual produce una inconsistencia.

El problema de asignación de calle consiste en realiza una correcta correspondencia entre la medición GPS del vehículo y la calle por la cual realizó la ruta. En el ejemplo anterior, se observa que asignando a la calle más cercana, el algoritmo falla.

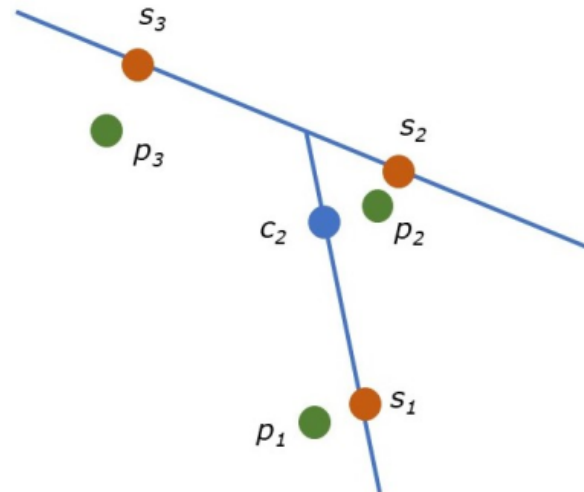


Figura 1: Imagen de una calle que es dividida en 2.

Fuente: “Pre-Processing Techniques for Improving a Topological Map-Matching Algorithm” de Blazquez, León, Pérez y Delgado [Blazquez et al., 2022].

2.2. Map-Matching Algorithms

Los algoritmos de correspondencia/emparejamiento de mapas o *Map-Matching Algorithms* (MMAs) buscan corregir ubicaciones y asociarlas a las calles o carreteras correspondientes.

Los MMAs han sido ampliamente estudiados, empleando diferentes tipos de modelos que trabajan en base a diversas técnicas de tipo geométrica sencilla, topológicos, estimación probabilística o de inferencia matemática compleja.

Debido a estos factores determinantes y modelos de trabajo distintos, en [Chao et al., 2020] se clasifican como:

- **Similarity Model:** Estos algoritmos entregan asignaciones de calles en base a la dirección y/o distancia entre la medición GPS obtenida y la trayectoria que se conoce del vehículo. Siguen intuitivamente la red de carreteras subyacente con la condición restrictiva de que el vehículo no puede cambiar de segmento en base a su trayectoria.

Dentro de este modelo se consideran 2 sub-modelos destacables:

- **Distance-based:** Son los modelos más primitivos y sencillos, como su nombre lo indica asigna una calle a la medición GPS en base a la distancia entre estas. Una distancia utilizada en muchos modelos es la distancia de Fréchet [Efrat et al., 2002], que en este contexto consiste en el cálculo de la distancia mínima que debe existir entre el punto anterior de asignación y la próxima asignación futura, esto mediante diversos cálculos [Bang et al., 2016]. Dada su simpleza en algunos casos, o robustez de estos, sus resultados pueden variar en distintos tipos de casos, también se pueden ver afectados por valores *outliers* ocasionando errores.

- **Pattern-based:** Estos modelos asignan calles en base a coincidencias históricas de asignaciones en los mapas mediante una puntuación, dado que trabajan con el supuesto de que las personas tienden a seguir las mismas rutas. Debido a la diversidad que puedan poseer los datos históricos, además del sesgo que puede existir entre estos, los resultados pueden resultar poco confiables y útiles al momento de asignar calles fuera de mapas con registros.
- **State-Transition Model:** Consiste en la construcción de un grafo que contiene las distintas probabilidades de asignación, las cuales poseen un peso de representatividad en base a la transición del vehículo entre los estados para así conseguir una ruta óptima de forma global.

Para la construcción del grafo existen 3 formas principales: *Hidden Markov Model* (HMM), *Conditional Random Field* (CRF) y *Weighted Graph Technique* (WGT).

- **Hidden Markov Model:** Es uno de los modelos más utilizados, trabaja creando estados ocultos que en base a observaciones se les corresponde una probabilidad a cada estado, por ende, se considera a todas las carreteras cercanas como una potencial asignación de ubicación del vehículo. Este tipo de modelo tiende a sufrir problemas de sesgo dado que considera solo las observaciones del estado oculto actual, además conlleva problemas de latencia.
- **Conditional random field:** A diferencia de los HMM, los CRF evitan el sesgo de considerar solo el estado oculto actual modelando el grafo en base al registro anterior de observaciones. A pesar de lo anterior trabaja modelos estadísticos similares a HMM y ambos terminan con poseer un problema de recuperación en caso de una desviación de la asignación correcta.
- **Weighted graph technique:** Este tipo de modelos consiste en inferir la ruta en base a un grafo de asignaciones candidatas ponderadas por pesos, los cuales son escogidos según su cercanía a la medición dentro de un radio de selección de candidatos. Siguen pasos similares entre los diferentes modelos de ponderación, pero esta es la principal diferencia entre los algoritmos dentro de esta clasificación.
- **Candidate-Evolving Model:** Este tipo de modelos funciona mediante un sistema de votos sistémicos, como medida de consideración de las posibles asignaciones a fin de mantener la consistencia no solo de puntos anteriores sino de asignaciones posteriores. Eliminando inconsistencias en una trayectoria votada en asignaciones posteriores, obteniendo así una evolución de resultados durante el proceso. Dentro de este tipo de modelos se presentan 2 soluciones alternativas.
 - **Particle filter:** Consiste en una técnica de estimación considerando una combinación de muestreo de Monte Carlo y la inferencia Bayesiana. Se utiliza para apoyar la asignación y en especial la corrección de errores anteriores en base a la estimación recursiva, donde las partículas con mayor peso cuentan con más posibilidades de propagarse, mientras se tiende a eliminar los casos con un peso menor.

- **Multiple hypothesis technique:** También busca mantener una serie de candidatos, pero no busca la recursividad, sino que las soluciones evolucionan a lo largo del tiempo durante las asignaciones, reduciendo cálculos futuros iterando en casos futuros.
- **Scoring Model:** Son diferentes algoritmos que no utilizan una ponderación de puntaje en particular. Los candidatos solo se escogen mediante la cercanía a la medición y son asignados a fin de maximizar la función de puntuación que tengan definida o criterios que previamente están definidos en el modelo.

Dado lo anterior, existen distintos algoritmos que abordan el problema de asignación de calles, con diferentes tipos de complejidad y obtención de resultados.

2.3. Topological Map-Matching Algorithms

Existe un subconjunto de algoritmos dentro de los MMAs que trabajan en base a las redes topológicas. En [Blazquez *et al.*, 2018] se recalca que los **TMMAs** son altamente utilizados para resolver el problema de ambigüedad espacial para los ITS dada la simplicidad de su funcionamiento, eficiencia y facilidad de su implementación. Estos algoritmos trabajan con el cálculo de caminos más cortos y/o con el tiempo de viaje entre pares de puntos GPS coincidentes, utilizando la topología de la carretera y las restricciones de giro para así asignar la calle a una medición GPS. Una ventaja de este tipo de algoritmos es la capacidad de entregar un resultado con una alta precisión de la posición sin la necesidad de utilizar grandes cantidades de memoria.

En [Leon *et al.*, 2020] y [Blazquez *et al.*, 2022] se proponen y analizan técnicas para la mejora de resultados para los TMMAs. Ambos toman enfoques diferentes, el primero busca implementar mejoras en base a un modelo post-procesado en el cual se agrega un paso para asignar a una calle las mediciones que en el algoritmo original no han logrado ser asignadas con éxito. En el segundo se propone un enfoque centrado en técnicas de pre-procesamiento antes de ejecutar un TMMA buscando mejorar la calidad de la solución y disminuir los tiempos de ejecución. La primera técnica determina un tamaño de *buffer* óptimo para una asignación potencial de calles, mientras que la segunda descarta posibles asignaciones con respecto a la dirección y el rumbo que posee el vehículo para así seleccionar posibles candidatos. Comparando así los resultados en un caso de prueba obtenido a través de mediciones en la comuna de Renca, Santiago, Chile.

En la Figura 2 se muestra el diagrama de flujo del MMA con el diseño de la secuencia de pasos. En primer lugar, los puntos GPS no están en las calles, por lo tanto no se puede calcular el camino más corto entre ellos. Se calcula la ruta más corta entre los puntos snap s_i y s_j , que son las proyecciones de los puntos GPS sobre las calles k_i y k_j . Si este camino no es factible, entonces el algoritmo obtiene candidatos alternativos de la línea central de la carretera dentro del **buffer** alrededor del punto de datos k_j . En consecuencia, se comprueban

las trayectorias viables entre el punto de datos k_i y el nuevo punto ajustado $alt k_j$. Si no se encuentran caminos viables, entonces se obtienen candidatos alternativos de la línea central de la carretera dentro del *buffer* alrededor del punto de datos k_i , y se recalculan los caminos más cortos y se comparan las velocidades entre los puntos vecinos inmediatos k_{i-1} y k_j . La variable booleana ficticia f_w ayuda a avanzar hacia atrás o hacia delante para probar trayectorias con puntos de datos consecutivos hasta que se resuelva el Problema de asignación de calle o se hayan probado N puntos de datos consecutivos.

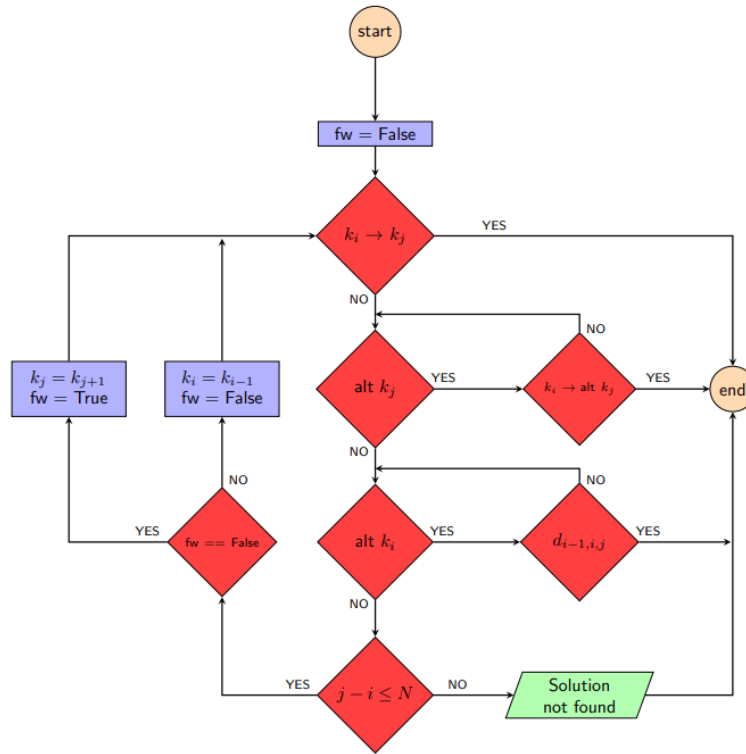


Figura 2: Diagrama de flujo de TMMA.

Fuente: “An Instance-Specific Parameter Tuning Approach Using Fuzzy Logic for a Post-Processing Topological Map-Matching Algorithm” Blazques, C., Ries, J., Miranda, P.A., León, R. J. [Blazquez et al., 2018].

2.4. Importancia de los parámetros en los MMAs

Los MMAs trabajan con diversos parámetros que se deben definir para su funcionamiento. La elección de estos parámetros juegan un rol importante en la calidad de los resultados obtenidos. En [Blazquez y Vonderohe, 2009] se presenta un estudio del efecto en el rendimiento al controlar los parámetros en los MMAs. Los parámetros estudiados son tres controlados por el usuario y dos externos obtenidos a través de datos simulados en un Decision-Rule Map-Matching algorithm. Se da un especial énfasis a los parámetros *buffer size*, *speed range*

tolerance y *number of consecutive data points*.

Buffer size corresponde al radio de distancia desde la cual se toman posibles calles candidatas para la asignación. Un valor muy pequeño de este parámetro causará que un menor número de calles sean escogidas aumentando así el rendimiento computacional. Por otra parte, se pueden descartar calles que no son se encuentran en la ruta real del vehículo, causando errores. Un valor alto de este parámetro, por el contrario, considerará un mayor número de calles como candidatas aumentando así la variabilidad y exploración del espacio de búsqueda, sin embargo, esto trae consigo una disminución de rendimiento computacional, requiriendo mayor tiempo de análisis. A pesar de lo anterior, el estudio mencionado no busca enfocarse en resultados de tiempos de ejecución.

Speed range tolerance es la tolerancia u holgura del cálculo de velocidad necesaria para que el vehículo recorra la distancia entre el punto asignado anteriormente a una calle al punto a asignarse en la calle candidata. Este punto se encuentra en la distancia mínima entre la medición GPS y la calle candidata de manera perpendicular entre sí, es decir la línea entre la calle y el punto que forma un ángulo de 90 grados. La relevancia de este parámetro radica en la eliminación o descarte de posibles calles candidatas si la velocidad requerida para que el vehículo viaje desde el punto anterior al punto posterior es muy alta, es decir, la distancia recorrida no es coherente. Un valor bajo de este parámetro puede causar que un alto número de calles sean descartadas o que todas las calles sean descartadas, como en una carretera con un vehículo que recorre grandes distancias en poco tiempo. Un alto valor puede causar que el filtro de la velocidad pierda relevancia al eliminar un bajo número de calles candidatas, requiriendo así un mayor número de iteraciones del algoritmo.

Number of consecutive data points corresponde a la cantidad de puntos consecutivos que se le entrega al algoritmo para calcular la viabilidad de una ruta de calles candidatas. El número de puntos asignados tiene su importancia dado que el problema no solo debe asignar que 2 puntos consecutivos sean coherentes, sino que la ruta en general posee una coherencia, es por ello por lo que cuando no se encuentra una calle viable para la ruta entre las calles candidatas se aumenta el número de puntos consecutivos para así ampliar el espacio de búsqueda y de esa forma encontrar nuevas rutas. Al igual que los parámetros anteriores, un valor bajo de este parámetro puede causar que el algoritmo no pueda corregirse si se asigna un punto de manera errónea obteniendo así una ruta general con fallas de coherencia entre sus calles. No obstante, aumentarlo en gran medida incrementará el número de rutas a escoger, aumentando el tiempo de ejecución.

En el estudio se pueden observar diferentes resultados en base al uso del algoritmo con datos obtenidos durante el invierno de 2002-2003 en Wisconsin e Iowa. En los resultados se puede observar que a mayor valor del *buffer size*, *speed range tolerance* y *number of consecutive data points* minimiza los falsos negativos, malas asignaciones y maximiza la cantidad de casos resueltos en comparación a valores de parámetros genéricos. Además muestra una convergencia a valores determinados de falsos negativos al aumentar *speed range tolerance* y *number of consecutive data points*. Al aumentar el valor de los parámetros, mejora la calidad del resultado entregado. Sin embargo, los resultados tienden a converger rápidamente,

por ello, aumentar los valores de los parámetros no traería consigo algún tipo de beneficio en cuanto a calidad, pero, afectará al rendimiento temporal del algoritmo, demostrando así la importancia de definir correctamente el valor de un parámetro para los MMAs, lo cual es el foco de este trabajo.

2.5. Algoritmos de sintonización

A continuación, se presenta una introducción a la sintonización de parámetros, el estado del arte y algunos algoritmos de sintonización con sus diferentes enfoques.

2.5.1. Sintonización de parámetros

La elección de valores para los parámetros de un algoritmo juega un papel de gran importancia, no solo porque cambia la calidad de las soluciones que entregan los algoritmos, sino que afectan igualmente al rendimiento computacional de estos.

En este trabajo se busca proponer una versión adaptativa del algoritmo topológico para la resolución de ambigüedad espacial, donde dada la variedad de situaciones a las que los **TM-MAs** se deben enfrentar, la definición de valores apropiados para los parámetros resulta ser muy compleja de generalizar. Es por lo anterior que se busca integrar una herramienta de sintonización de parámetros para así obtener conocimiento del ajuste de parámetros dentro de los **TMMAs** haciendo que este último sea capaz de adaptarse a diferentes situaciones y entregar así resultados satisfactorios.

Para entrar en contexto de la utilidad de las herramientas de sintonización de parámetros, se debe comprender qué es el ajuste de parámetros. El ajuste de parámetros o sintonización es un proceso donde se controla la configuración de los valores de los parámetros del algoritmo con el fin de optimizar diferentes aspectos de este, y tal como se presentó en la sección 2.4, la sintonización de los parámetros del problema puede afectar drásticamente los resultados a obtener por un algoritmo.

La sintonización de parámetros es un proceso de ajuste o afinamiento de los parámetros, los cuales no pueden ser aprendidos directamente de los datos y deben ser definidos con anterioridad para el algoritmo. Estos procesos de sintonización son, muchas veces, técnicas utilizadas en el aprendizaje automático y la optimización con el objetivo de encontrar una combinación de parámetros que optimizan el valor de la función objetivo conseguida por un algoritmo, mejorando así la calidad de la solución esperada.

2.5.2. Estado del arte

Para lograr la sintonización de parámetros se utilizan los algoritmos sintonizadores, que corresponden a métodos que aplican diferentes técnicas o modelos para realizar la búsqueda automatizada de los parámetros a través de diferentes técnicas y modelos de trabajo.

En [Montero *et al.*, 2014] se presenta un estudio introductorio que recopila diferentes técnicas de algoritmos sintonizadores, definiendo y categorizando algunos métodos de ajustes de parámetros.

Esta clasificación de métodos de ajuste son:

- **Hand-made tuning:** Método típico de trabajos de sintonización más antiguos. Consiste en que el diseñador estudia el rendimiento del algoritmo en diferentes instancias del problema. Utilizando parámetros escogidos bajo su criterio y que son modificados de forma iterativa ajustando los valores a mano, como su nombre lo indica, mientras se consigan mejoras.
- **Tuning by analogy:** Estos métodos trabajan en base a pautas o estudios de otros autores realizados anteriormente para así ajustar los parámetros del algoritmo actual a situaciones conocidas que han mostrado buen desempeño.
- **Experimental Design Based Tuning:** Se consideran algoritmos categorizados en este tipo de métodos los que utilizan técnicas basadas en diseño de experimentos para el ajuste de parámetros.
- **Search Based Tuning:** Método basado en el ajuste de parámetros conforme se explora el espacio de búsqueda de los parámetros de algoritmos en las diferentes iteraciones.
- **Hybrid Tuning:** Utiliza una combinación de métodos para ajustar parámetros.

En el mismo estudio se presentan cuatro algoritmos sintonizadores. Estos son:

- **F-Race:** Algoritmo presentado en [Birattari *et al.*, 2002] en el año 2002. Implementa una metodología basada en los *Racing Methods*, los cuales están clasificados dentro de los *Experimental Design Based Tuning*. Consiste en un método iterativo que busca proporcionar una mejor configuración de parámetros en base al rendimiento de un conjunto definido de candidatos. En cada iteración los candidatos actuales se van evaluando de manera de determinar quiénes son estadísticamente peores y deben abandonar la carrera en cada paso.

F-Race está especialmente adaptado para afinar conjuntos de parámetros para algoritmos de búsqueda estocástica. Utiliza el *Friedman Two-Way Analysis of Variance by Ranks* para comparar el rendimiento de las configuraciones candidatas en cada paso. Dado que el método está implementado en un diseño de bloques, que considera

como variación las diferentes instancias del problema, que toma como hipótesis nula “todas las clasificaciones de los candidatos dentro de cada bloque son considerados igualmente probables”.

F-Race puede resultar de gran utilidad en situaciones donde no se tenga acceso a una gran cantidad de recursos computacionales, dada la forma de priorizar configuraciones más prometedoras, especialmente usado en sintonizar parámetros en problemas de búsqueda estocástica.

- **Revac:** Es un algoritmo evolutivo, dentro de la clasificación *Search Based Tuning*, basado en la teoría de la información para medir la relevancia de sus datos [Nannen y Eiben, 2007]. Sus operadores evolutivos se utilizan para determinar la distribuciones de los parámetros. El conjunto de configuraciones de parámetros a sintonizar es llamado una población. Se inicia con una población en que, para todos los parámetros, se considera una distribución uniforme. Conforme se avanza, los valores iniciales se verán afectados por operadores de transformación, reduciendo el rango de valores de los parámetros que muestran buen desempeño. Se consideran parámetros útiles si estos cuentan con un bajo nivel de entropía, obteniendo así una mayor relevancia frente a sus pares respecto a los resultados que obtiene el algoritmo sintonizado utilizando los mejores valores de parámetros.
- **SPO:** o *Sequential Parameter Optimization* es un método iterativo que realiza un análisis experimental de un conjunto de configuraciones de parámetros. Estima el rendimiento de un algoritmo mediante un modelo de proceso estocástico para así estimar nuevas configuraciones prometedoras que permitan, a su vez, refinar el modelo. Comienza construyendo un conjunto de configuraciones iniciales mediante un muestreo de hipercubos por sobre un rango de valores de los parámetros definidos.
- **ParamILS:** Este último al ser un algoritmo utilizado en la propuesta de solución será presentado en detalle en la sección 2.5.4.

2.5.3. Evoca

En [Riff y Montero, 2013] se propone el Calibrador Evolutivo conocido también como **Evoca**. Evoca es un algoritmo sintonizador que se presenta como una herramienta para personas no experimentadas con el diseño de metaheurísticas. El algoritmo está diseñado con la capacidad de obtener resultados equivalentes frente a algoritmos de sintonización más complejos sin recurrir a la necesidad de tener un conocimiento amplio sobre el problema a resolver.

El funcionamiento como calibrador se puede observar en la Figura 3. En esta figura se muestra el proceso de diseño iterativo de Evoca. Inicialmente, se tiene un conocimiento bajo o nulo del problema. Los resultado de las heurísticas se evalúan para determinar si existe una mejora; de no ser así, se repite el ciclo rediseñando el modelo con lo aprendido en la iteración anterior. El ciclo finaliza cuando se determina que los resultados del algoritmo son

satisfactorios. La implementación del modelo busca obtener resultados tempranos del ajuste gracias a Evoca, evaluarlos, y, si es necesario, rediseñar la implementación del modelo o integrar un sintonizador de mayor complejidad, hasta finalmente entregar una versión de desempeño aceptable del algoritmo.

Definición: Dado una algoritmo A y sus parámetros categóricos $P_c = c_1, \dots, c_k$ y parámetros numéricos $P_n = n_1, \dots, n_m$ que pertenecen a los dominios $\Theta = \Theta_1, \dots, \Theta_{m+k}$, el objetivo es encontrar los valores para los conjuntos P_c y P_n tales que estos valores permitan al algoritmo obtener su mejor rendimiento.

La representación del cromosoma en Evoca corresponde a un *string* donde cada elemento corresponde a un parámetro, y su valor se asigna dentro del dominio del parámetro correspondiente. Por lo tanto, la longitud del *string* es el número de parámetros que deben ser sintonizados en el proceso.

Para generar la población inicial se debe tener en cuenta el número de parámetros a utilizar y sus tamaños de dominios iniciales. Es clave incluir todos los valores permitidos para cada parámetro, de forma independiente en la primera población. Para parámetros categóricos o de dominios discretos esto no produce ningún tipo de inconveniente. No obstante, para datos de tipo numérico se debe dividir en intervalos cuya extensión se deberá ajustar según las necesidades de precisión del diseñador.

Dada su naturaleza evolutiva, Evoca utiliza dos operadores. El primero de ellos es un operador de cruce *Wheel-Crossover*, el cual genera un hijo tomando para cada gen un valor seleccionado aleatoriamente desde todos los valores presentes en la población actual. Posteriormente el hijo reemplaza al peor individuo de la población. El segundo operador corresponde a un operador de mutación, que modifica un valor de los parámetros del hijo, generado por el operador de cruce, mediante *Hill Climbing First Improvement*. Esta mutación está obligada a realizarse y en casos de valores numéricos se selecciona mediante una elección aleatoria dentro de un rango continuo. El nuevo individuo reemplaza al segundo peor individuo de la población.

Finalmente en el algoritmo 1 se puede observar el funcionamiento de Evoca. El cual comienza con una población inicial en la línea 1, que ingresa al ciclo de la evolución de la población, ciclo que terminará solamente si se cumple el criterio de termino de la línea 12, que puede estar relacionado a algún criterio de convergencia de calidad o tiempo de ejecución. Dentro del ciclo de evolución se producen dos tipos de búsqueda de mejoras dentro de la población, la primera es a través de cruzamiento de la población como se muestra en la línea 4, manteniendo las mejores características de la población y entregándoselas a un nuevo hijo. Una segunda búsqueda se realiza a través de una mutación del hijo del paso anterior como se muestra en la línea 7. Las nuevas soluciones reemplazan a los dos peores individuos de la población actual.

En [Rojas-Morales y Riff, 2023] se realiza un estudio del rendimiento al utilizar **OS-EVOCA**, extensión de Evoca que aprende de las configuraciones que producen un efecto positivo o

Algorithm 1 EVOCA

Require: target-algorithm, P_c , $precision_c$, P_n , Θ **Ensure:** Parameter Values

- 1: Generate Initial Population
 - 2: **repeat**
 - 3: New Population $\leftarrow$ Current Population
 - 4: Child $\leftarrow$ Wheel-crossover(Population)
 - 5: Evaluate Child using R random seeds
 - 6: Replace the worst chromosome by Child in New Population
 - 7: Child-mutated $\leftarrow$ Mutation(Child)
 - 8: Evaluate Child-mutated using R random seeds
 - 9: **if** Child-mutated is better than Child **then**
 - 10: Replace the second worst chromosome by Child-mutated in New Population
 - 11: **end if**
 - 12: **until** termination_criterion
-

negativo en una evaluación de las configuraciones que entrega al sintonizador. **OS-EVOCA** tiene la peculiaridad que a través de esta evaluación es capaz de modificar y entregar nuevos conjuntos iniciales de configuraciones a Evoca, lo cual busca mejorar los resultados del algoritmo objetivo en base a una mejor selección inicial de configuraciones de parámetros previo al ajuste de estos utilizando una mejor cantidad de recursos computacionales.

2.5.4. ParamILS

En [Hutter *et al.*, 2007] se presenta el primer algoritmo de configuración de parámetros basado en la búsqueda local iterada. El algoritmo se presenta como un acercamiento capaz de sintonizar los parámetros de algoritmos deterministas como algoritmos de naturaleza estocástica, pudiendo ser aplicado de forma independiente del escenario del problema a ajustar y del objetivo del proceso de optimización.

Para comprender el funcionamiento del método **ParamILS** se deben considerar ciertas definiciones de los conceptos del problema de forma de coincidir con las dadas por los autores.

Un algoritmo a optimizar A y la distribución de las instancias del problema a optimizar I . Cada uno de los parámetros del algoritmo se asocia a un dominio de posibles valores, este dominio se define como Θ_i . El espacio completo de configuraciones de parámetros de un problema puede entonces definirse como $\Theta = \Theta_0 \times \Theta_1 \times \dots \times \Theta_n$, la cual corresponde a todas las posibles combinaciones entre los valores posibles de los parámetros. El objetivo en esta situación es encontrar la configuración $\theta \in \Theta$ de forma de optimizar $A(\theta)$.

Para encontrar un “mejor” rendimiento, se trabaja en base a una suposición de un coste escalar dependiendo del escenario aplicado, $sc(A, \theta, I, s)$, de una única ejecución de $A(\theta)$

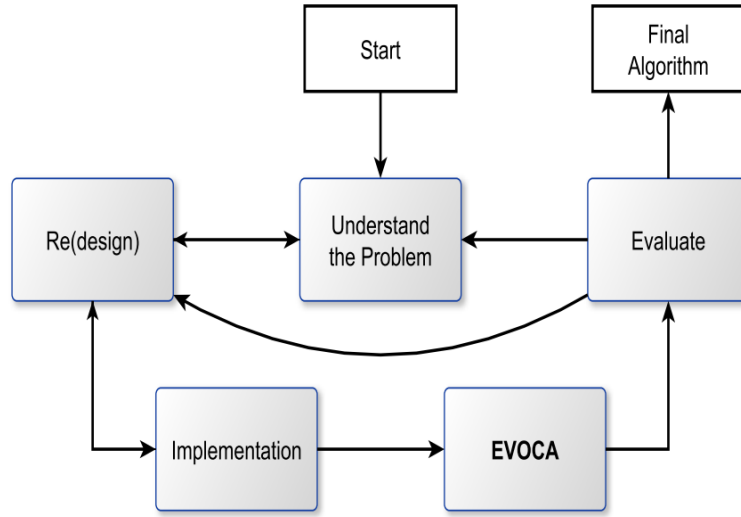


Figura 3: Diagrama de integración de Evoca.

Fuente: “A New Algorithm for Reducing Metaheuristic Design Effort” María-Cristina Riff and Elizabeth Montero [Riff y Montero, 2013].

en una instancia I procedente de una semilla s en caso de ser un algoritmo aleatorio. Este coste puede ser, el tiempo de ejecución, el error de aproximación o la mejora obtenida por sobre un coste de referencia en específico. Dado lo anterior, si se posee una distribución entre las instancias se puede obtener una distribución de costes $CD(A, \theta, I)$.

Por lo tanto, el objetivo es minimizar un estadístico, $c(\theta)$, de esta distribución de costes, siendo así un problema de optimización estocástico basándose en la aproximación de $c(\theta)$ de N muestras $\hat{c}_N(\theta)$

En el algoritmo 2 se puede ver el pseudo-código correspondiente a ParamILS. En un inicio se requiere una configuración del espacio del dominio Θ , θ_0 , un vecindario de relaciones N y la función *better* la cual compara el rendimiento de dos configuraciones en la línea 1. El *output* esperado es la mejor configuración de parámetros θ encontrada durante la ejecución del algoritmo como se muestra en la línea 2.

Al iniciar se buscan configuraciones aleatorias, θ dentro de Θ comparando con la original, θ_0 , si resulta mejor, θ reemplaza a θ_0 . Esto se muestra entre las líneas 4 y 9.

Posteriormente ejecuta la función *IterativeFirstImprovement* (θ, N) que recibe como parámetros θ_0 y N . Esta función es la encargada de obtener una primera mejora de θ de forma aleatoria dentro del espacio $N(\theta')$, donde θ' corresponde a θ que se entrega en un inicio. Asignando la configuración retornada en θ_{ils} como se muestra en la línea 10.

Es así como el algoritmo comienza un ciclo **while** que se ejecuta mientras el criterio de término no se haya cumplido. Este ciclo se compone de tres etapas, una etapa de perturbación,

donde se obtiene θ de forma aleatoria. La etapa de búsqueda local, la cual usa la función *IterativeFirstImprovement* para obtener una mejora de θ con respecto a la solución inicial. La última etapa de verificación del criterio de aceptación, haciendo uso de la función *better* compara los θ encontrados y si resultan ser mejores que θ_{ils} estos reemplazan a θ_0 . Además dentro del ciclo existe la probabilidad p_{restart} de que se realice un cambio aleatorio dentro de θ_{ils} . Si el criterio de aceptación se cumple, se termina el ciclo, retornando así la configuración de parámetros θ_0 que corresponde a la mejor configuración encontrada.

En nuevas propuestas de algoritmos, se utiliza ParamILS en la definición de configuraciones iniciales de parámetros, acelerando la obtención de resultados para su posterior análisis. En [Szczepanski et al., 2020] se busca proponer un nuevo algoritmo enfocado en paralelización de arquitecturas multinúcleo de problemas combinatorios, con MO-ParamILS, una versión extendida de ParamILS.

2.5.5. Irace

En [López-Ibáñez et al., 2016b], los autores presentan *The Irace Package* donde se presenta una extensión del método **F-Race**. Este método extendido se trata de un método del tipo de carreras iteradas para la configuración automática.

El proceso para encontrar una configuración viene dado por tres pasos principales:

- Muestreo de nuevas configuraciones acorde una distribución determinada.
- Selección de las mejores configuraciones muestreadas mediante carreras.
- Actualización de la distribución de muestreo para sesgarlo hacia las mejores configuraciones.

Los pasos anteriores se repiten hasta lograr cumplir con algún criterio de término. Cada parámetro configurable tiene asociada una distribución de muestreo que es independiente del resto de parámetros. La naturaleza de estas distribuciones varía. Se emplea una distribución normal truncada para los parámetros numéricos o una distribución discreta en el caso de los parámetros categóricos.

Los parámetros ordinales se consideran equivalentes a valores numéricos. La actualización de las distribuciones consiste en modificar la media y la desviación típica en el caso de la distribución normal, o los valores de probabilidad discretos de las distribuciones discretas. La actualización sesga las distribuciones para aumentar la probabilidad de muestrear, en futuras iteraciones, los valores de los parámetros en las mejores configuraciones encontradas hasta el momento.

El Algoritmo 3 presenta el pseudo-código correspondiente al algoritmo Irace y su esquema de carreras iteradas. La carrera iterada requiere como entrada un conjunto de instancias I

Algorithm 2 ParamILS - BasicILS

```

1: Input: Parameter configuration space  $\Theta$ , neighbourhood relation  $N$ , procedure better
   (compares  $\theta, \theta' \in \Theta$ )
2: Output: Best parameter configuration  $\theta$  found.
3:  $\theta_0 \leftarrow$  default parameter configuration  $\theta \in \Theta$ 
4: for  $i \leftarrow 1 \dots R$  do
5:    $\theta \leftarrow$  random  $\theta \in \Theta$ 
6:   if better( $\theta, \theta_0$ ) then
7:      $\theta_0 \leftarrow \theta$ 
8:   end if
9: end for
10:  $\theta_{ils} \leftarrow$  IterativeFirstImprovement ( $\theta_0, N$ );
11: while not TerminationCriterion() do
12:    $\theta \leftarrow \theta_{ils}$ 
13:   —Perturbation—
14:   for  $i \leftarrow 1 \dots s$  do  $\theta \leftarrow$  random  $\theta' \in N(\theta)$ 
15:   end for
16:   —LocalSearch—
17:    $\theta \leftarrow$  IterativeFirstImprovement ( $\theta, N$ );
18:   —AcceptanceCriterion—
19:   if better( $\theta, \theta_{ils}$ ) then
20:      $\theta_0 \leftarrow \theta$ 
21:   end if
22:   with probability  $p_{restart}$  do  $\theta_{ils} \leftarrow$  random  $\theta \in \Theta$ 
23: end while
24: return overall best  $\theta$  found
25: Procedure IterativeFirstImprovement ( $\theta, N$ ):
26: repeat
27:    $\theta' \leftarrow \theta$ 
28:   foreach  $\theta'' \in N(\theta')$  in randomised order do
29:     if better( $\theta'', \theta'$ ) then
30:        $\theta \leftarrow \theta''$ ; break
31:     end if
32: until  $\theta' = \theta$ 
33: return  $\theta$ 

```

muestreadas de $\mathbb{I}$, un espacio de parámetros X , una función de coste C y un presupuesto de ajuste B .

Como se puede observar en 3, en un primer paso, la carrera comienza en una configuración inicial candidata Θ_1 la cual se genera a través del muestreo del espacio de parámetros X que se recibe como entrada en la línea 1. Con las configuraciones candidatas muestreadas, se da inicio a una carrera, en la cual las configuraciones se evalúan en la instancia haciendo uso de la función de costos, para así realizar una separación de las configuraciones que funcionan de peor forma que otras pudiendo ser eliminadas si existe una relación estadística entre las configuraciones que llevan a los peores resultados en la línea 2. Esta eliminación puede ser llevada a cabo según diferentes filtros, como los presentados en 2.5.2 en el ítem de **F-Race**, no obstante, Irace implementa las pruebas t-pareadas en lugar de la prueba de Friedman, además incorpora varias mejoras, como el muestreo a partir de una distribución normal truncada, una implementación paralela, una estrategia de reinicio que evita la convergencia prematura y un procedimiento de carrera elitista para garantizar que las mejores configuraciones de parámetros encontradas también se evalúan en el mayor número de instancias de entrenamiento.

El resto de configuraciones que continúan pasan a ser Θ^{elite} que volverán a ser partícipes de las siguientes instancias desde la línea 4 hasta la línea 8. En las instancias posteriores se generan nuevas configuraciones mediante un padre y una probabilidad entregada según un rango definido por la suma de los rangos y el coste medio de la configuración obtenidos de instancias anteriores. Las configuraciones con un mayor rango tendrán una probabilidad más alta de ser padres y que nuevas configuraciones hereden sus características, esto es lo que se realiza dentro de la función $Sample(X, \Theta^{elite})$.

Cuando ya sean generadas las nuevas configuraciones se realiza una nueva carrera y se repite el mismo proceso hasta agotar el presupuesto de ajuste inicial B (dentro del *while* iterador de la línea 4) obteniendo así las configuraciones sobrevivientes encontradas.

Algorithm 3 Irace

Require: $I = \{I_1, I_2 \dots\} \sim \mathbb{I}$, parameter space : X , cost measure: $C(\theta, i) \in \mathbb{R}$, tuning budget: B

- 1: $\Theta_1 = SampleUniform(X)$
- 2: $\Theta^{elite} = Race(\Theta_1, B_1)$
- 3: $j = 1$
- 4: **while** $B^{used} \leq B$ **do**
- 5: $j = j + 1$
- 6: $\Theta^{new} = Sample(X, \Theta^{elite})$
- 7: $\Theta^{elite} = Race(\Theta_j, B_j)$
- 8: **end while**
- 9: **Output:** Θ^{elite}

En [Menezes *et al.*, 2020] se propone un método de ajuste automático de parámetros para

el algoritmo **RatSLAM**, algoritmo que busca solucionar el problema de la robótica móvil de localización y mapeo simultáneos (**SLAM**), el cual utiliza Irace para encontrar los mejores ajustes de parámetros.

Otra situación donde se utiliza Irace como algoritmo sintonizador de parámetros se estudia en [Prestes *et al.*, 2018], donde se utiliza dentro de **Multi-Objective Evolutionary Algorithm based on Decomposition with Dynamical Resource Allocation (MOEA/D-DRA)**, el cual cuenta con un gran éxito, pero debido a que cuenta con un gran número de parámetros y operadores con los cuales trabajar, hacen que su selección sea una tarea difícil de realizar.

2.6. Resumen del Capítulo

En este capítulo se aborda el Problema de asignación de calles, cuyo origen está fuertemente relacionado con la precisión de los sistemas GPS, que proporcionan mediciones no completamente precisas. Esta falta de precisión dificulta la correcta asignación de calles o carreteras a los vehículos, generando inconsistencias para algoritmos simples de asignación.

Además, se exploran los algoritmos **MMAs** y **TMMAs** como métodos capaces de abordar eficazmente esta problemática de asignación, habiéndose destacado por sus buenos resultados en la literatura. La problemática central de este trabajo tiene que ver con la configuración óptima de parámetros para estos algoritmos. Definir de manera óptima estos parámetros manualmente es difícil, si no imposible, dada la cantidad de tiempo que requiere este estudio y la influencia significativa que tienen los valores de los parámetros en los resultados, como se ha evidenciado en diversos estudios.

En este contexto, se examinan diferentes algoritmos sintonizadores como herramientas para la configuración de parámetros. Estos sintonizadores permiten determinar de manera efectiva los parámetros de los algoritmos de este estudio. Aunque existen varias opciones de sintonizadores, se realiza un análisis detallado de los algoritmos EVOCA, ParamILS e Irace para evaluar y comparar sus resultados en este trabajo.

CAPÍTULO 3

PROPUESTA DE SOLUCIÓN

En el capítulo 2.4 se presenta la relevancia de los parámetros *buffer* (**radio de selección de calles**) y *tol* (**tolerancia de velocidad**) en la obtención de una alta exactitud en la asignación de puntos en el mapa dentro del **TMMA**, siendo el valor utilizado para evaluar este proceso es con la *exactitud*, que corresponde al valor retornado por el **TMMA** el cual pertenece al intervalo $[0,1]$, donde valores más altos o cercanos a 1 son considerados como una asignación de calles más exacta. Sin embargo, la tarea de llegar a estos valores óptimos es compleja y dada la diversidad de escenarios en la que se pueden encontrar los datos, no existe una regla general para su selección.

En este trabajo se propone un acercamiento para la predicción de parámetros de los **TMMAs** que permita adaptar los valores de los parámetros de radio de selección de calles (*buffer*) y tolerancia de velocidad (*tol*) a diferentes conjuntos de datos provenientes de las mediciones GPS de los vehículos. El estudio considera fijar los valores de dichos parámetros minimizando la exactitud (*accuracy*) de la asignación de calle de los **TMMAs**. Para lograr esta predicción, se utilizarán algoritmos sintonizadores capaces de encontrar un ajuste de parámetros a cada instancia de datos GPS y posteriormente el uso de un modelo de regresión para buscar predecir valores de *buffer* y *tol* basados en los resultados de los procesos de sintonización.

3.1. Proceso de trabajo

La Figura 4 ilustra el proceso de trabajo en diferentes etapas. La primera etapa requiere un conjunto de datos iniciales (rectángulo amarillo), que incluye datos de mediciones GPS, un conjunto de valores para el *buffer* y un conjunto de valores para *tol*, necesarios para los sintonizadores.

La primera etapa, representada en color celeste, consiste en la sintonización de parámetros para el **TMMA** utilizando los conjuntos iniciales de parámetros. Los resultados de esta sintonización se entregan a la siguiente etapa (color rojo), que involucra la creación y experimentación de modelos de regresión lineal para desarrollar un **TMMA** adaptativo utilizando estos modelos de predicción lineal.

Los algoritmos sintonizadores obtienen resultados de sus experimentos al ejecutar múltiples veces el algoritmo a ajustar, y es por esta razón que deben tener acceso al **TMMA**, además deben contar con diferentes datos GPS (C_{GPS}) para su ejecución. Los resultados de estas ejecuciones, *accuracy*, son utilizados por los algoritmos sintonizadores para así tener una estimación de la calidad de sus configuraciones. Se procesarán y analizarán los resultados de cada sintonizador, donde sus salidas corresponderán a una serie de pares *buffer* y *tol* con mayor rendimiento en su respectivo caso.

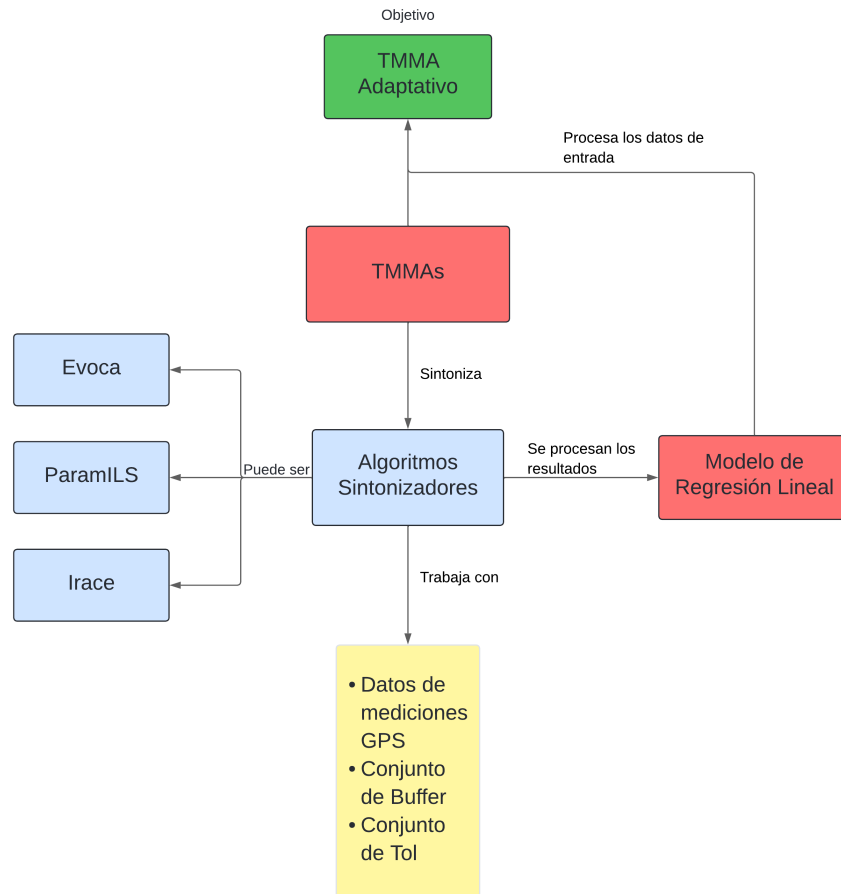


Figura 4: Diagrama de flujo de trabajo.

Fuente: Elaboración propia

Se utilizará la sintonización de parámetros sobre el **TMMA** con el fin de afinar los parámetros *buffer* y *tol* según un conjunto inicial de estos, C_{buffer} y C_{tol} , previamente investigados dentro de los registros previos del uso de los **TMMAs** para limitar el trabajo de los sintonizadores a resultados dentro de rangos de calidad. Es importante indicar que cada algoritmo sintonizador trabajará de forma independiente.

Las configuraciones de parámetros obtenidas serán utilizadas como los valores reales a obtener en la construcción de diferentes modelos de regresión lineal. Con los cuales predecir nuevas configuraciones frente a diferentes datos GPS sin tener conocimientos previos de estos.

Estos modelos serán evaluados en cuanto a rendimiento de sus resultados dentro del **TMMA** y su costo de ejecución, dado que se busca proponer una solución cuya penalización sea menor que ejecutar múltiples veces el **TMMA** con diferentes valores para los parámetros *buffer* y *tol* con el propósito de encontrar un buen resultado.

Finalmente, los modelos con mejor desempeño se incorporarán como una mejora al algoritmo **TMMA** como un proceso que pretenderá adaptar los parámetros de entrada, *buffer* y *tol*, frente a casos de pruebas nuevos.

3.1.1. ArcGIS

El software ArcGIS se utiliza para visualizar y explorar los datos de los Sistemas de Información Geográfica, lo que permite mapear y asignar manualmente los puntos GPS a calles según su posición geográfica dentro de los mapas de la aplicación. Sus extensiones como ArcMap permiten trabajar desde una interfaz gráfica, lo que a su vez permite ver los datos de entrenamiento y las asignaciones de calles que realiza el algoritmo **TMMA**. Además contiene el módulo *arcpy* de Python, el cual es utilizado por el algoritmo para acceder a los mapas [Esri, 2012].

Se hace mención a este software por lo esencial que resulta para la ejecución del **TMMA** y dado que es licenciado, condiciona a trabajar en el sistema operativo Windows.

3.1.2. Parámetros a ajustar

Se seleccionaron dos conjuntos de parámetros críticos para el **TMMA**:

- *buffer*: corresponde al radio de distancia que conforma una circunferencia, cuyo centro es el punto GPS a asignar, desde la cual seleccionar las diferentes calles candidatas para la asignación de este punto, determinando así el área de influencia del proceso de mapeo que realiza el **TMMA**.

El conjunto de valores para el *buffer*, cuyas unidades son *foot* (ft), utilizado dentro del proceso de sintonización fue creado en base valores utilizado en otros estudios como [Blazquez *et al.*, 2018], además de agregarle otros valores diferentes a los encontrados, utilizando de base los definidos, para que de esta forma escapar de una definición fija de valores y entregar la posibilidad de que los algoritmos sintonizadores configuraciones más variadas. Los valores seleccionados son:

$$C_{\text{buffer}} = \{b_0, b_1, \dots, b_{15}\} \text{ con } b_i = (60 + 20i)[\text{ft}]$$

- *tol*: corresponde a un umbral máximo de velocidad a considerar si la distancia recorrida de un punto GPS es válida dependiendo de la velocidad medida en el punto a considerar y en el punto anterior, asegurando que las mediciones reflejen movimientos realistas del vehículo.

Al igual que para el *buffer*, el conjunto de valores para *tol*, cuyas unidades son *mi/la/hora* (mi/hr), fue creado en base valores utilizado en otros estudios acerca de los

MMA, además de agregarle otros valores diferentes a los encontrados, buscando una mayor variabilidad de configuraciones posibles. Los valores seleccionados son:

$$C_{\text{tol}} = \{10, 15, 20, 25, 30\}[\text{mi/hr}]$$

3.2. Fundamentos del uso e implementación de algoritmos sintonizadores

En este trabajo se utilizan algoritmos sintonizadores con el objetivo de afinar parámetros para los **TMMA**. Los datos obtenidos a partir de los sintonizadores permitirán generar recomendaciones de valores de parámetros basados en las características de los nuevos casos de pruebas y su similitud con datos que se utilizaron en el proceso de entrenamiento del algoritmo adaptativo.

Se trabajará con diferentes algoritmos sintonizadores, los cuales utilizarán las mismas instancias de entrenamiento y rangos de valores para los parámetros. No obstante, cada uno buscará una respuesta en base a su metodología de optimización particular, evaluando el rendimiento de los parámetros seleccionados y ajustando sus estrategias conforme a los resultados obtenidos. Este enfoque permitirá comparar la eficacia de cada sintonizador en diversas condiciones y con distintos conjuntos de datos.

Para realizar la sintonización de parámetros se utilizarán tres algoritmos diferentes, **ParamILS**, **Evoca** e **Irace**. A continuación se describe en detalle la metodología de cada caso.

3.2.1. Evoca

La elección de este algoritmo sintonizador es la simpleza de su implementación y forma de iteración evolutiva a través de cruzamiento y mutación como se exploró en 2.5.3. Dada la facilidad de implementación, Evoca resulta ser una buena opción como primer acercamiento a trabajar con sintonizadores sin saber demasiado del funcionamiento de estos algoritmos, tal y como recomiendan sus autoras.

La aplicación de **Evoca** en este trabajo requirió adaptar algunas componentes desde un sistema Linux hacia un sistema Windows. Por lo anterior, se modificó el código fuente de Evoca para ejecutarlo dentro del ambiente Windows. Fuera de estos cambios solo se modificó el criterio de Evoca de siempre buscar un óptimo que maximizara el resultado de los **TMMA**, además de entregarle los conjuntos de parámetros a sintonizar y las instancias de ejecución.

La configuración utilizada para Evoca se realizó con el fin de obtener la mejor configuración de ambos parámetros en cada instancia. Otra de sus configuraciones a considerar, es que el algoritmo solo permite el uso de números enteros dentro del valor mínimo y el valor máximo

especificados dentro de los conjuntos mostrados en la sección 3.1.2, es decir, su selección es discreta dentro del conjunto pero no posee la selección de valores fijos. Este tipo de selección permitió experimentar con una mayor variación de configuraciones de parámetros posibles. Las configuraciones restantes se mantuvieron por defecto.

Los parámetros que sintonizará, y sus conjuntos iniciales, son presentados en la sección 3.1.2, y dado que la configuración de selección de parámetros de Evoca no permite la selección discreta de valores, es por ello que se configuró para seleccionar valores dentro del intervalo [60,360] para *buffer* y del intervalo [10,30] para *tol*.

3.2.2. ParamILS

La elección de ParamILS como algoritmo sintonizador en este trabajo se justifica por sus buenos resultados demostrados en estudios anteriores dentro de la literatura. Apoyándose en lo descrito en la sección 2.5.4, el algoritmo es un sintonizador con un nivel mayor de complejidad a comparación con otros algoritmos. Dada su estructura robusta basada, un proceso de comparación de configuraciones que utiliza un criterio de dominio, se espera obtener buenos resultados a partir de sus ajustes de parámetros.

Este algoritmo sintonizador se configuró con la intención de maximizar el valor retornado por el **TMMA**, la *exactitud*, teniendo acceso completo a los conjuntos de valores de parámetros de 3.1.2 requiriendo sintonizar ambos parámetros en cada paso del sintonizador. Cuenta con un tiempo máximo de ejecución de aproximadamente de una semana y media por cada instancia de datos. Otras configuraciones se mantienen por defecto a fin de no afectar de manera considerable el funcionamiento del algoritmo a la hora de experimentar.

Los parámetros que sintonizará, y sus conjuntos iniciales, son presentados en la sección 3.1.2, y dada la configuración de ParamILS, solo seleccionara valores dentro de los conjuntos iniciales de estos parámetros.

3.2.3. Irace

La elección de Irace se justifica por su diferencia respecto a los dos algoritmos sintonizadores ya presentados: Evoca y ParamILS. Su mayor diferencia viene dada por el proceso de eliminación de configuraciones, esto en base a pruebas estadísticas que están implementadas en el algoritmo .

La configuración de Irace, se realizó con el fin de maximizar el valor de la *exactitud* en base a una selección elitista de parámetros, es decir, buscará según sus criterios solo configurar *buffer* o *tol*, esto con el fin de experimentar con la mantención de uno de los valores estáticos en las instancias de datos. Sus configuraciones restantes se mantuvieron por defecto.

Los parámetros que sintonizará, y sus conjuntos iniciales, son presentados en la sección 3.1.2, y dada la configuración de Irace, solo seleccionara valores dentro de los conjuntos iniciales de estos parámetros

3.2.4. Metodología de Experimentación

La metodología de experimentación realizada se puede observar en la Figura 5, donde cada sintonizador se ejecutó de forma independiente con los cinco conjuntos de datos GPS. Se realizaron múltiples iteraciones para cada combinación de parámetros *buffer* y *tol*, recopilando los resultados de *exactitud* para cada ejecución.

3.3. Implementación de la solución

Posterior a los procesos de sintonización de parámetros en el **TMMA** en las diferentes instancias de datos, se buscará crear modelos de regresión lineal, en base a la evaluación de cada una de las configuraciones de valores de los parámetros *buffer* y *tol* obtenidos en cada sintonización. Así entonces, se agruparán los resultados de *buffer* de un sintonizador y se construirá un modelo de regresión para predecir este parámetro. El proceso se repetirá para el parámetro *tol* y por cada sintonizador a utilizar.

El proceso anterior se puede observar en la Figura 5, donde para N instancias de datos se utilizan los algoritmos Evoca, ParamLS e Irace, representados por los colores azul, naranja y verde respectivamente. Estos algoritmos se emplean por separado para obtener una configuración de *buffer* y *tol* necesarias para el **TMMA** a partir de los conjuntos iniciales de estos. Las configuraciones se separan por parámetro y por algoritmo sintonizador; por ejemplo, $\text{Evoca}_{\text{buffer}1}$ representa el valor obtenido por Evoca en la instancia 1 de datos. Estos valores se reúnen con los del resto de configuraciones resultantes de las diferentes instancias de datos, generando nuevos conjuntos que se utilizan como los valores esperados en los diferentes modelos de regresión lineal. Los modelos continúan diferenciándose tanto por el algoritmo sintonizador involucrado como por el parámetro que buscan predecir. Esta separación se realiza para comparar los resultados obtenidos por el algoritmo adaptativo que utiliza estos diferentes modelos, con el objetivo de identificar cuál modelo y algoritmo sintonizador son los mejores, y así compararlos con los resultados del **TMMA** que utiliza valores determinados por la literatura.

3.3.1. Modelos de Regresión

La decisión de trabajar con modelos de regresión lineal se justifica por su facilidad de interpretación, permitiendo entender la relación entre las variables para generar la salida mediante coeficientes directos. Similar a lo anterior, se determina utilizar modelos de regresión

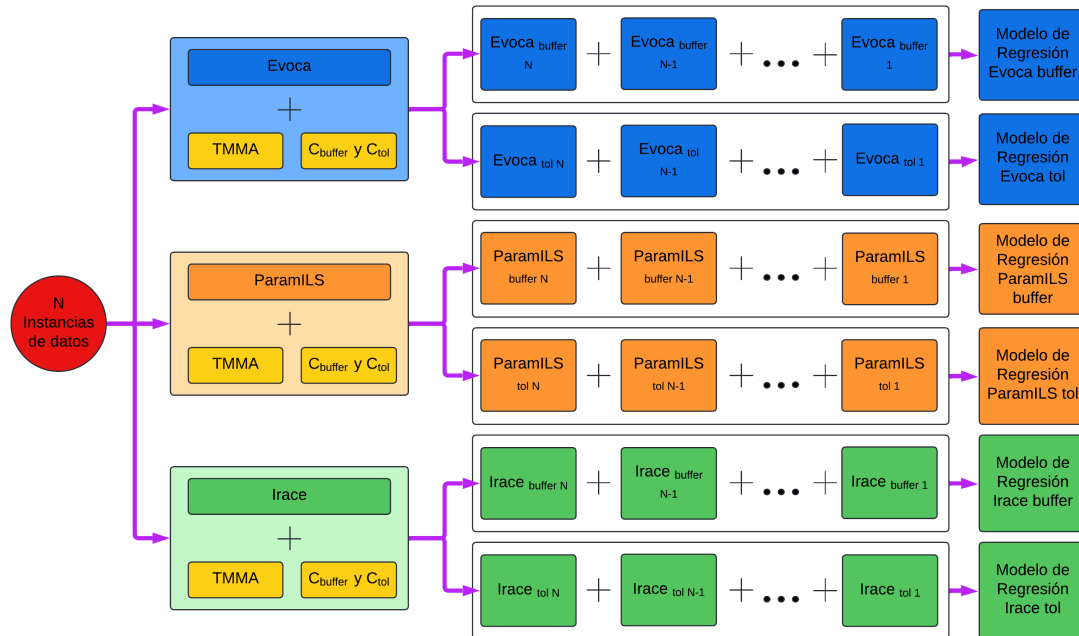


Figura 5: Diagrama del proceso de generación de modelos de regresión usando la data de los procesos de sintonización.

Fuente: "Elaboración propia"

separados tanto para *buffer* como para *tol*, esto, con el fin de estudiar el comportamiento y relación que poseen de forma independiente entre sí con respecto a los datos entrantes.

Además, la generación de modelos de regresión lineal es eficiente en términos de recursos computacionales, permitiendo un veloz entrenamiento y rápida una predicción. Esto es ventajoso especialmente al comparar su uso frente a simplemente ejecutar algoritmos sintonizadores en cada paso del proceso. La ejecución de los algoritmos **TMMAs** conllevan un alto costo de ejecución y, por consecuencia, no sería recomendable ejecutar un método que requiera múltiples llamadas del algoritmo para solo conseguir un único par *buffer* y *tol* útiles. Los cuales podrían no ser los adecuados para ciertas situaciones, requiriendo un mayor costo computacional al volver a reproducir el proceso anterior para encontrar nuevos valores.

Así entonces, en este trabajo se plantea una análisis de los componentes de las mediciones GPS registradas en los archivos de entrada de los TMMAs para ser consideradas características de este tipo de modelos. En este caso se han estudiado transformaciones de dichas características bien conocidas en la literatura con el propósito de mejorar la calidad de los modelos generados.

La posterior incorporación de estos modelos al **TMMA** se evaluará en base al comportamiento de estos dos recursos en conjunto. Este proceso se describe en mayor detalle en el siguiente capítulo. Es importante destacar que estos modelos no alterarán los datos entrantes de mediciones GPS, manteniéndolos en su estado natural antes de ser entregados al

TMMA diferenciándose así de un proceso de pre-procesamiento de datos. La metodología propuesta puede ser definida como un análisis de componentes previos utilizando modelos de regresión para adaptar los hiper-parámetros con los que trabajan los TMMA.

3.4. Resumen

En este capítulo se presenta una propuesta de solución que busca la mejora de los algoritmos **TMMAs**, determinando los parámetros *buffer* y *tol* según los datos GPS de los archivos de entrada. Para esto se utilizan los algoritmos sintonizadores: Evoca, ParamILS e Irace, los cuales se justifican y explican en detalle. Por último, se describe la implementación de modelos de regresión lineal dentro de la solución. Se justifica su elección considerando su eficiencia computacional y facilidad de uso para predecir los parámetros siendo entrenados a partir de los resultados de los algoritmos sintonizadores. Esto, con la finalidad de mejorar la *exactitud* en la asignación de calles con los parámetros *buffer* y *tol* que se estimarán de los modelos, y cuyos resultados deben ser comparados con una configuración obtenida desde la literatura.

CAPÍTULO 4

VALIDACIÓN DE LA SOLUCIÓN

El propósito de este capítulo es validar la efectividad de la solución propuesta para la determinación automática de los valores de los parámetros *buffer* y *tol* en el algoritmo **TMMA** basado en las características de datos GPS de cada caso de prueba a considerar. Para ello, se han realizado experimentos de sintonización de parámetros elitista, de selección discreta dentro de un conjunto y de selección continua en un intervalo utilizando valores enteros; frente a múltiples datos GPS para entrenamiento para modelos, y una variedad de experimentos de predicción de parámetros, utilizando diferentes modelos de regresión lineal, para su uso dentro del **TMMA**. Se han analizado los resultados obtenidos, con el fin de demostrar que las predicciones realizadas, provenientes de los modelos de regresión lineal, obtienen resultados con altos niveles de *exactitud*, que puede tomar valores en el conjunto $[0, 1]$, frente a casos desconocidos de pruebas.

La fase de experimentación se divide en dos secciones diferentes, la primera fase consistirá en realizar pruebas con los algoritmos sintonizadores y comparar su rendimiento frente al **TMMA** para ajustar los valores de los parámetros *buffer* y *tol*. Para esto se utilizará un conjunto de valores para cada uno de los parámetros en diferentes casos de prueba seleccionados con anterioridad.

La segunda fase consistirá en experimentar con la creación de diferentes modelos de regresión, los cuales se entrenarán con los resultados entregados por los sintonizadores y los datos GPS de entrada utilizados en la sintonización. Estos datos GPS serán pre-procesados utilizando algunas propiedades de los datos, tales como, el promedio, la desviación estándar, entre otros, de columnas cuyos valores que corresponden a la distancia, velocidad o tiempo. El cálculo de componentes, de ciertas columnas que componen a los datos, se realiza con el fin de generar una variedad de modelos de predicción para su evaluar de *exactitud* frente a diversos casos de prueba.

Se finaliza la validación con una comparación de resultados de ejecución del **TMMA** adaptativo creado en base al mejor modelo de regresión lineal seleccionado con valores encontrados en la literatura como valores generales.

4.1. Experimentos de Algoritmos Sintonizadores

Este proceso de experimentación con los algoritmos sintonizadores está enfocado en determinar una configuración adecuada de valores para los parámetros *buffer* y *tol* y así, mejorar la exactitud de la asignación de puntos GPS a las calles del algoritmo objetivo de estos sintonizadores, el **TMMA**.

4.1.1. Instancias de Datos

Para este proceso de experimentación, se utilizaron cinco instancias diferentes de datos GPS, Tabla 1, provenientes de diferentes rutas de vehículos de la ciudad de Portage, Wisconsin, Estados Unidos, priorizando la diferencia entre estas rutas y la cercanía de calles que tienen unas con otras. Dado que hay datos divididos según la continuidad de toma de mediciones GPS, se decide utilizar datos que posean mediciones cada diez segundos para utilizar la mayor cantidad de puntos dentro del **TMMA**. Estas instancias pueden ser observadas dentro de un mapa en la Figura 6.

Instancias de Datos
1140268103
1150268101
1160268102
1160268107
1190268102

Tabla 1: Instancias de Datos Usados en la Sintonización y Entrenamiento de Modelos de Regresión.

4.1.2. Entorno de Ejecución

El algoritmo sintonizador ParamILS fue configurado en el lenguaje de programación Ruby versión 2.5.3.

El algoritmo sintonizador Evoca fue modificado para adaptarlo al ambiente de trabajo utilizando el lenguaje C++ dentro de un compilador g++ versión 4.8.2.

El algoritmo sintonizador Irace fue utilizado en su versión 3.5.6 y fue ejecutado como librería desde el lenguaje de programación R 4.3.3.

El código del **TMMA** es construido y ejecutado utilizando el lenguaje de programación Python versión 2.7.18.4.

La aplicación de licencia ArcMAP, para la visualización y obtención de información geográfica, se utiliza en su versión 10.8.2.

Los experimentos fueron realizados en un computador con un sistema operativo Windows 11 Pro, el cual cuenta con un procesador Intel(R) Core(TM) i7-8559U CPU @2.70 Hz y 8GB de memoria RAM.

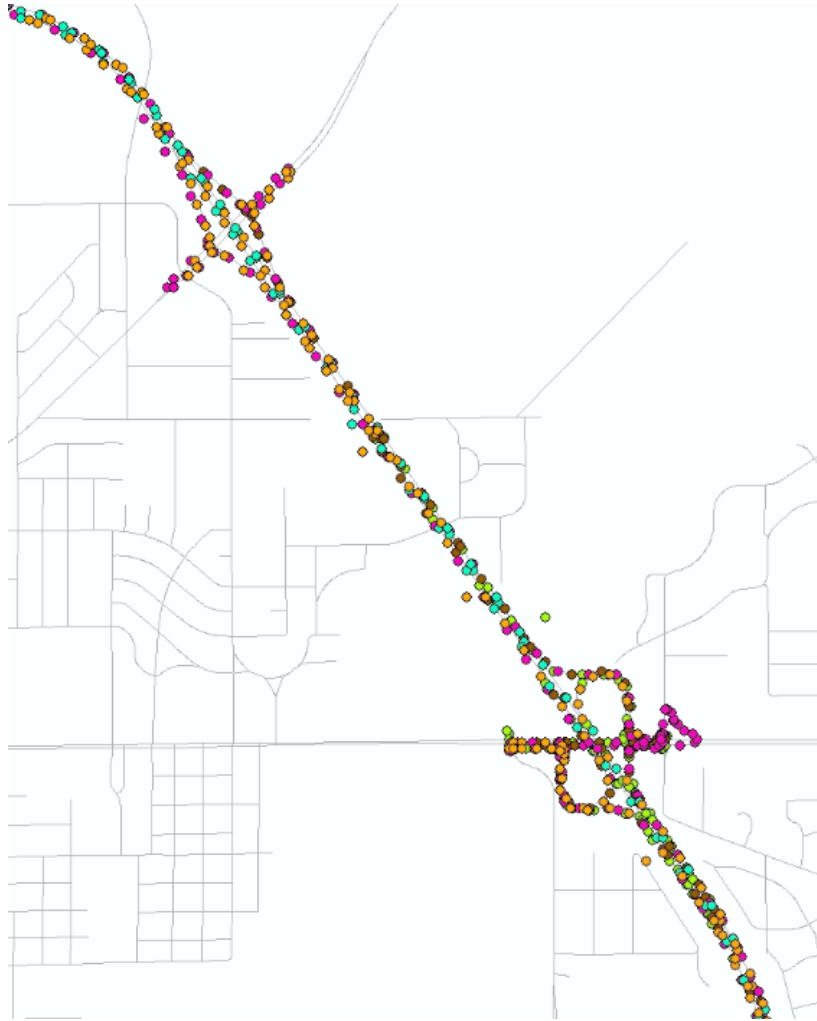


Figura 6: Mapa de Los Puntos GPS en el Mapa de ArcMap.
Fuente: "Extracto de ArcMap"

4.2. Resultados de Sintonización

Una vez realizadas las sintonizaciones de parámetros, se obtuvieron una variedad de resultados suficientes para continuar con la siguiente etapa de experimentación. A continuación, se presentan los resultados de este proceso de sintonización, un análisis individual del desempeño de cada sintonizador, un análisis a la experimentación y una comparación de los resultados.

Los resultados entregados están empaquetados en una lista, la cual contendrá el valor de los parámetros *buffer*, *tol* y el correspondiente valor de *exactitud* que obtendrá el **TMMA** utilizando esa configuración.

4.2.1. Mejores Soluciones

En la Tabla 2 se muestran los resultados entregados por cada sintonizador en las diferentes instancias. Se lista, para cada conjunto de datos y sintonizador los valores obtenidos para *buffer* y *tol*. Además, para cada configuración de parámetros se muestra el valor de *exactitud* que obtiene el **TMMA** al utilizarla.

Set de Datos	ParamILS			Evoca			Irace		
	<i>buffer</i>	<i>tol</i>	<i>exactitud</i>	<i>buffer</i>	<i>tol</i>	<i>exactitud</i>	<i>buffer</i>	<i>tol</i>	<i>exactitud</i>
1140268103	280	30	0.895	315	27	0.895	240	10	0.491
1150268101	180	30	0.952	360	30	0.952	160	10	0.583
1160268102	60	30	0.888	360	30	0.888	120	10	0.551
1160268107	360	20	0.875	360	30	0.875	60	10	0.679
1190268102	180	30	0.818	180	25	0.798	120	10	0.596

Tabla 2: Resultados de los Sintonizadores.

La Figura 7 muestra el desempeño de los resultados de los algoritmos sintonizadores en cada instancia.

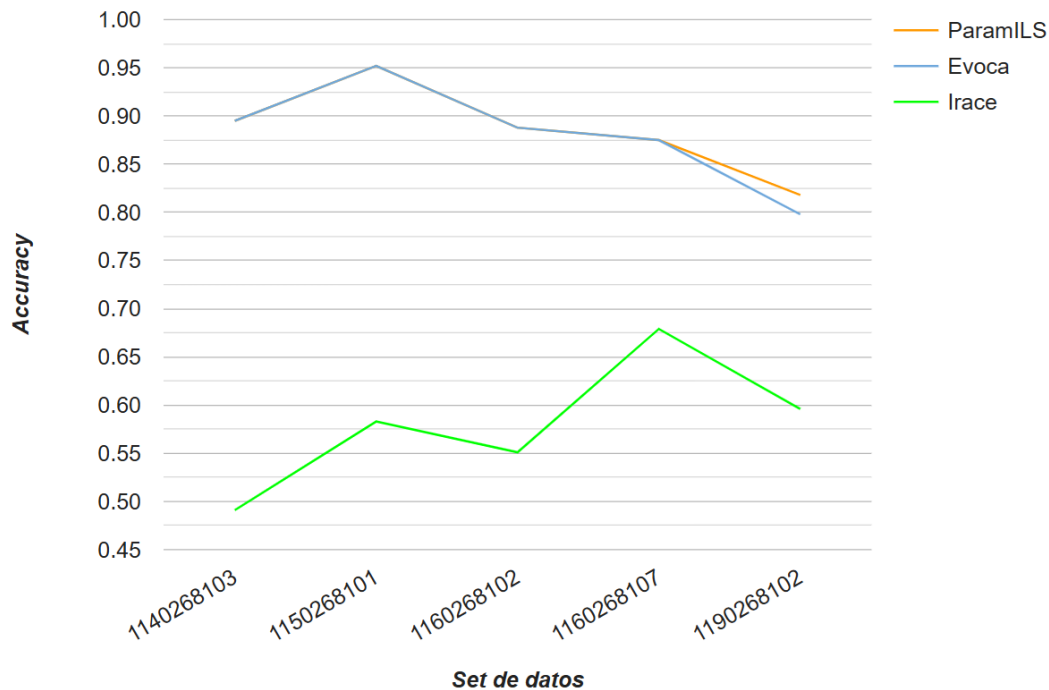


Figura 7: Gráfica exactitud vs Conjunto de Datos.

Fuente: Elaboración propia

Se observa que diferentes configuraciones de parámetros pueden conseguir el mismo valor de *exactitud*, como se puede observar en las configuraciones obtenidas por ParamILS y Evoca para los conjuntos de datos 11402681033, 1150268101, 1160268102 y 1160268107, en los cuales 2 de estas 4 instancias, para 1150268101 y 1160268102, se repitió para ambas configuraciones el valor de 30 para *tol*.

En ciertos conjuntos de datos la configuración de *buffer* o *tol* puede tener una mayor predominancia a la hora de condicionar. No obstante, no significa que un solo parámetro tenga la totalidad del peso de la solución, esto se puede observar en los resultados obtenidos por Irace, el cual en todas sus ejecuciones sesgó su ajuste de parámetros al *buffer* dejando siempre fijo el valor de *tol*. Así, debido a la selección elitista dentro de su algoritmo, obtuvo valores de *exactitud* mucho más bajos en comparación de los otros sintonizadores. Esto se retomará en la sección 4.2.2.

ParamILS y Evoca obtuvieron buenos resultados de *exactitud*, obteniendo los mismos valores para cuatro de los cinco conjuntos de datos utilizados. ParamILS obtiene un mejor resultado en una de las instancias, tal y como se esperaba dada la literatura.

Evoca, por otra parte, tiende a entregar en sus configuraciones, valores de *buffer* más altos que el resto de sintonizadores, esto es importante mencionarlo debido a que el *buffer* corresponde al valor del radio de circunferencia al considerar las calles candidatas a asignación. Un alto valor del *buffer* corresponderá a una mayor serie de comprobaciones dentro del **TMMA** pudiendo tener cierto impacto en el tiempo de ejecución, no obstante, su impacto no debería ser tan alto dentro de los posibles valores del conjunto que pueden ser seleccionados, los cuales están definidos en la sección 3.1.2.

4.2.2. Otras Configuraciones de Parámetros

En la sección anterior se mostraron las mejores configuraciones entregadas como salida de los procesos de sintonización, pero como se mencionó también, diferentes configuraciones son capaces de obtener el mismo valor de *exactitud* u otros resultados altos pueden ser útiles de mencionar de cara a la siguiente etapa de experimentos.

Con el fin de ampliar esta idea de que diferentes configuraciones pueden entregar la misma *exactitud*, en la Tabla 3 se muestran otras dos mejores configuraciones, de una gran cantidad de configuraciones encontradas, de los sintonizadores ParamILS y Evoca, las configuraciones de Irace se omiten dada la baja calidad de sus resultados. Como se puede observar en la tabla, estas configuraciones tienden a no disminuir la calidad de los resultados e incluso para el caso de ParamILS las configuraciones tienen una *exactitud* igual que la mejor solución encontrada, pero no utiliza la misma configuración de parámetros.

En la Tabla 3 se puede observar un pequeño aumento de variedad en la configuración de *tol* pero aún predominando el valor de 30 dentro de los resultados de ParamILS cuyos valores

Set de Datos	ParamILS			Evoca		
	<i>buffer</i>	<i>tol</i>	<i>exactitud</i>	<i>buffer</i>	<i>tol</i>	<i>exactitud</i>
1140268103	280	25	0.882	255	23	0.882
	340	25	0.882	240	22	0.882
1150268101	260	30	0.952	330	28	0.946
	140	30	0.952	270	24	0.935
1160268102	100	30	0.888	330	28	0.867
	80	30	0.888	315	27	0.860
1160268107	360	25	0.875	255	23	0.866
	360	30	0.875	240	22	0.866
1190268102	140	30	0.818	195	25	0.798
	100	30	0.818	135	25	0.798

Tabla 3: Segundas y Terceras Mejores Configuraciones.

no difieren de los mejores encontrados, pudiendo considerarse este valor crítico para estos conjuntos de datos. Esto podría acarrear un error en el entrenamiento de los modelos de regresión lineal, sesgo de clase desequilibrada, de la siguiente etapa de experimentos en cuanto sean evaluados en series diferentes. A pesar de lo anterior, ambos algoritmos muestran la capacidad de adaptarse a diferentes conjuntos de datos demostrando cierta estabilidad en los resultados que entregan, siendo esta ligeramente mayor en el caso de ParamILS.

4.3. Modelos de Regresión Lineal

En esta sección tendrá lugar la segunda etapa de experimentación de esta memoria, experimentación con modelos de regresión lineal a partir de los resultados obtenidos por los algoritmos sintonizadores.

Para obtener buenos resultados del proceso de regresión lineal es fundamental definir debidamente la entrada de los modelos que se desean crear. Las columnas coincidentes en las diferentes mediciones, y las más relevantes dentro del algoritmo **TMMA**, son las columnas **X**, **Y**, **Speed** y **Time**, donde esta última se calculó en segundos para utilizarla en los modelos y se renombró como **Time_seconds**.

Ahora bien, dada la disparidad de filas que pueden tener los datos se decide utilizar diferentes funciones para calcular indicadores representativos a partir de las columnas seleccionadas.

Las funciones seleccionadas para el cálculo de componentes son: la función máximo (Max), la función mínimo (Min), la función promedio (Mean) y la función para calcular la desviación estándar (STD). Estas elecciones están orientadas a equilibrar la complejidad de los métodos con la simplicidad y velocidad esperadas de la solución propuesta en este trabajo.

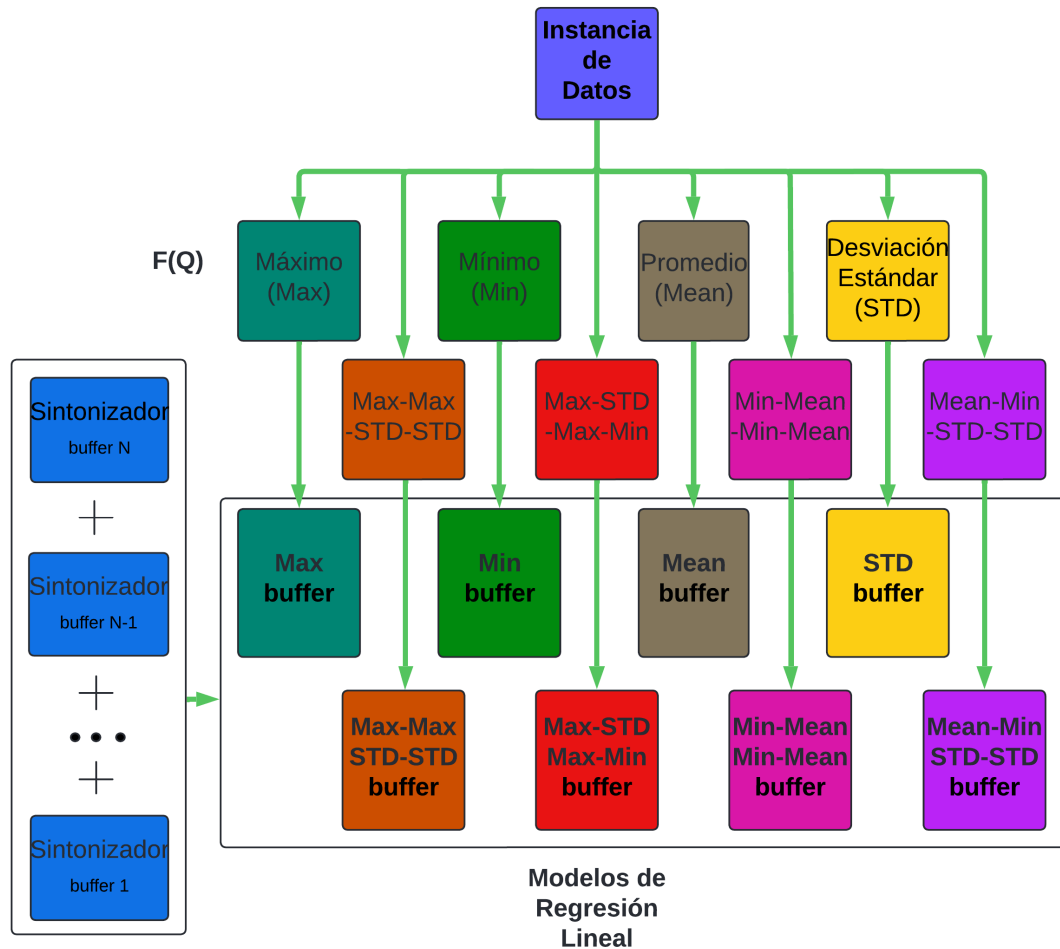


Figura 8: Diagrama de Modelos con el Cálculo de Componentes según sus Funciones.

Fuente: Elaboración propia

Con las funciones definidas, se plantea la creación de ocho diferentes modelos, en base a la formas del cálculo de las componentes de las columnas de datos GPS. De estos ocho modelos, en cuatro de ellos, todas las componentes fueron calculadas con la misma función, es decir, se creó un modelo solo con componentes calculadas con el máximo (Max) de las columnas, otro modelo con componentes calculados con el mínimo (Min), y así. Los cuatro modelos restantes, que se definen como $F(Q_i)$ donde $i \in \{1, 2, 3, 4\}$, corresponden al uso de las funciones anteriores pero aplicadas a diferentes columnas, estas combinaciones son: $Min_Mean_Min_Mean (F(Q_1))$, $Max_STD_Max_Min (F(Q_2))$, $Mean_Min_STD_STD (F(Q_3))$ y $Max_Max_STD_STD (F(Q_4))$, la selección de las variantes de funciones fueron elegidas al azar dentro de todas las combinaciones posibles que se podían formar utilizando las cuatro funciones originales. Por ejemplo, si se utiliza $F(Q_3)$ la componente correspondiente a la columna **X** es calculada con la función Mean, la componente correspondiente de la columna **Y** con la función Min, y de las columnas **Speed** y **Time** se obtienen las componentes con la función STD. Dada la alta cantidad de combinaciones posibles y con el fin de

evaluar algunos escenarios se seleccionaron de forma aleatoria sin priorizar el uso de una función por sobre otra.

Lo anterior se puede observar de mejor forma en la Figura 8, donde se presenta como ejemplo la creación de cualquier modelo de regresión lineal para la predicción del parámetro *buffer* según los resultados de los algoritmos sintonizadores.

La ecuación 4.1 representa los valores que serán entregados para el entrenamiento de los modelos de regresión lineal $m_{func,param,sint}$.

$$(\mathbf{x}_{func}, \vec{y}_{param,sint}) \Rightarrow m_{func,param,sint} \quad (4.1)$$

$$\mathbf{x}_{func} = \begin{pmatrix} C1_{1,func} & C2_{1,func} & C3_{1,func} & C4_{1,func} \\ C1_{2,func} & C2_{2,func} & C3_{2,func} & C4_{2,func} \\ C1_{3,func} & C2_{3,func} & C3_{3,func} & C4_{3,func} \\ C1_{4,func} & C2_{4,func} & C3_{4,func} & C4_{4,func} \\ C1_{5,func} & C2_{5,func} & C3_{5,func} & C4_{5,func} \end{pmatrix} \quad (4.2)$$

Donde $\mathbf{x}_{func}$ corresponde a la matriz de componentes y cada fila contiene los componentes de una serie de datos GPS de las cinco disponibles, calculadas utilizando una función *func*, la cual puede corresponder a las funciones *Max*, *Min*, *Mean*, *STD*, $F(Q_1)$, $F(Q_2)$, $F(Q_3)$ y $F(Q_4)$. Dado que se utilizarán cuatro columnas de estas series de datos GPS, sus componentes pueden ser definidas como:

- *C1*: componente calculada a partir de la columna **X**.
- *C2*: componente calculada a partir de la columna **Y**.
- *C3*: componente calculada a partir de la columna **Speed**.
- *C4*: componente calculada a partir de la columna **Time**.

$$\vec{y}_{param,sint} = (y_{1,param,sint}, y_{2,param,sint}, y_{3,param,sint}, y_{4,param,sint}, y_{5,param,sint})^T \quad (4.3)$$

La ecuación 4.3 muestra la salida $\vec{y}_{param,sint}$ correspondiente a los valores esperados de cada fila que contiene $\mathbf{x}_{func}$. El vector $\vec{y}_{param,sint}$ contiene las configuraciones de un parámetro, obtenidas por un sintonizador, el subíndice *param* corresponde al parámetro del cual se obtendrá la configuración, siendo estos *buffer* y *tol*, y el subíndice *sint* corresponde al sintonizador que obtuvo la configuración del parámetro, siendo estos Evoca, ParamILS e Irace. Las diferentes configuraciones de parámetros corresponden a las mejores configuraciones entregadas por los sintonizadores en la Tabla 2.

Según a las definiciones anteriores, $m_{func,param,sint}$ corresponde al modelo de regresión lineal cuyas componentes fueron calculadas por la(s) función(es) *func*, el parámetro objetivo es *param* y el algoritmo sintonizador que lo obtuvo *sint*.

Según lo anterior, al poseer dos parámetros que configurar, ocho tipos de funciones para el cálculo de componentes y tres sintonizadores encargados de obtener las configuraciones de parámetros se obtuvieron un total de 48 tipos de modelos diferentes. En esta etapa de experimentación se realizaron ocho pruebas por cada par de modelos según el tipo de sintonizador para cada instancia de datos GPS de pruebas, esto debido a que el **TMMA** requiere de la predicción de los dos parámetros configurados para ejecutarse manteniendo la homogeneidad de estos.

4.3.1. Instancias de Datos

Para este proceso de experimentación y validación, se utilizaron cinco conjuntos de datos diferentes contando con treinta instancias diferentes de datos para la ejecución, dado que cada conjunto de datos cuenta con tres series obtenidas cada 30 segundos, dos series obtenidas cada 20 segundos y una serie con la totalidad de los datos, estos conjuntos de instancias pueden encontrarse en las Tablas 10 y 11. Los datos GPS son provenientes de diferentes rutas de vehículos de la ciudad de Portage, Wisconsin, Estados Unidos.

Se utilizan instancias de datos de la misma procedencia que los datos utilizados para el entrenamiento de los modelos de regresión lineal, dada la accesibilidad que se posee a esta data. Las instancias obtenidas cada 10 segundos pueden ser observadas en la Figura 9.

4.3.2. Modelos de Regresión Lineal

Se cuenta con un total de 48 modelos de regresión lineal diferentes con el objetivo de predecir configuraciones de parámetros para el **TMMA**. Debido a que cada modelo fue entrenado con cuatro componentes, son considerados modelos de regresión lineal múltiple. En las Tablas 12 y 13, se muestran los coeficientes e intercepto de cada modelo según el algoritmo sintonizador y función(es) involucradas en su entrenamiento.

En esta etapa los modelos de regresión lineal tienen la tarea de predecir la configuración de los parámetros *buffer* y *tol* próximos a usar dentro del **TMMA**.

4.3.3. Entorno de Ejecución

El entorno de ejecución, además de utilizar los de elementos mencionados en la sección 4.1.2, se utiliza la librería scikit-learn versión 0.20.4 el entrenamiento de los modelos de re-

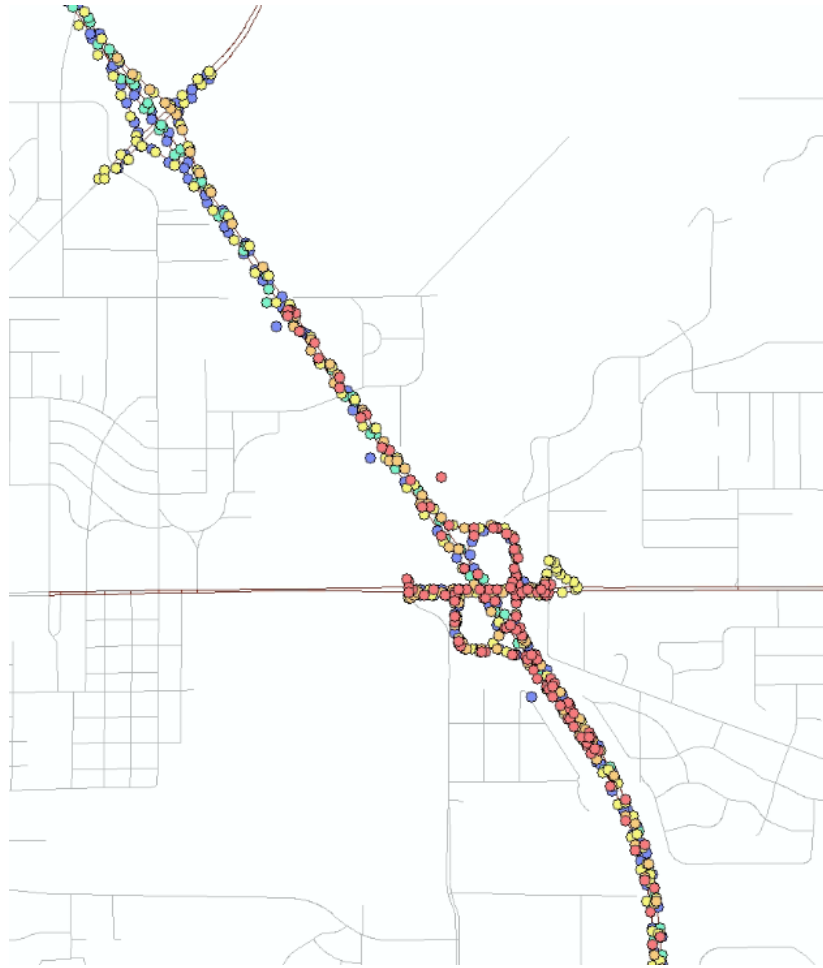


Figura 9: Mapa de Los Puntos GPS en el Mapa de ArcMap.

Fuente: "Extracto de ArcMap"

gresión lineal creados en el lenguaje de programación Python.

4.3.4. Metodología de Experimentación

La metodología de experimentación realizada con los modelos de regresión puede ser observada en la Figura 10, en la cual se puede apreciar el flujo de trabajo. Al inicio, se tiene una instancia de datos de prueba. A las columnas **X**, **Y**, **Speed** y **Time_seconds** se le aplica la función "Func" para así obtener los componentes necesarios de cara a utilizar los modelos de regresión, los cuales están entrenados, con los componentes procesados de los datos de entrenamiento por la función correspondiente, y cuya configuración fue encontrada por su respectivo algoritmo sintonizador.

Se entregan los componentes a los modelos para así obtener una configuración de parámetros a partir de la predicción, $buffer_c$ y tol_c , estos parámetros se utilizan dentro del **TMMA**

en la posterior ejecución de este, en la instancia de datos de prueba elegida en un inicio. De este proceso se espera obtener como retorno un valor de *exactitud* el cual corresponde al valor que se busca maximizar.

Lo anterior se hace con el objetivo de poder evaluar los modelos de regresión cuyos resultados esperados, a partir de sus datos de entrenamiento, fueron anteriormente obtenidos a través de los algoritmos sintonizadores. Pudiendo así tener una métrica de validación de la solución que se propone en esta memoria, además, se utilizaron diferentes funciones en el cálculo de componentes con el objetivo de encontrar una relación dentro de los datos que recibirá el **TMMA** y con la configuración adecuada de parámetros *buffer* y *tol*.

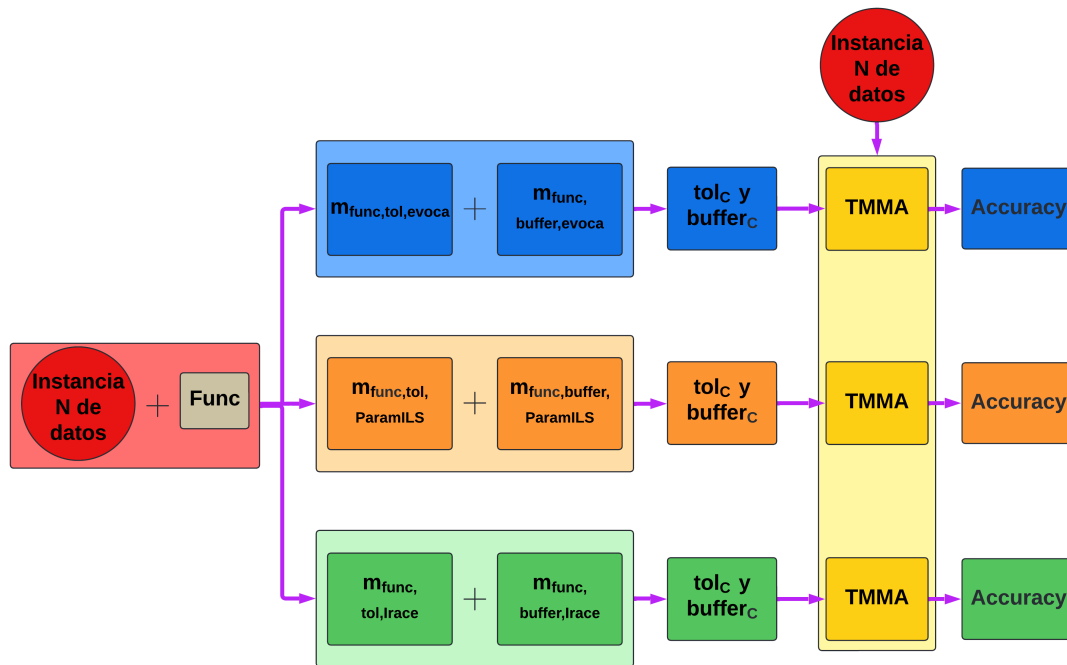


Figura 10: Diagrama Representativo de Experimentos de Modelos de Regresión Lineal.
Fuente: Elaboración propia.

4.4. Resultados de Experimentos con Modelos de Regresión

Una vez realizados los experimentos de predicción de parámetros *buffer* y *tol* utilizando diferentes modelos de regresión lineal para su posterior uso dentro del **TMMA**, se obtuvieron una variedad de resultados de *exactitud*. A continuación se presentan los resultados de este proceso de pruebas, un análisis individual y comparativo de los modelos según el método de construcción utilizado en cada uno de ellos y una conclusión final de los resultados.

Dada la variedad de instancias y modelos, los resultados obtenidos pueden ser observados en los diferentes gráficos de dispersión en las Figuras 11 y 13- 19 que muestran los valores

de *exactitud* obtenidos según el tipo de función utilizada para calcular los componentes de los datos utilizados en el entrenamiento de los modelos dentro de las diferentes instancias de pruebas. Esto se puede observar, por ejemplo, en la Figura 11, donde se encuentran los valores obtenidos por los modelos utilizando una función de ejemplo. La función mostrada es Min y sus valores objetivos fueron anteriormente sintonizados por ParamILS, Evoca e Irace.

Los gráficos de dispersión restantes han sido agregados al capítulo **ANEXOS** en la sección 5.3.

4.4.1. Resultados Según Función

Dada la variedad de funciones que se utilizaron en el proceso de construcción de modelos, se decide mostrar un resumen de los datos obtenidos en las Tablas 4 y 5 según el tipo de función utilizada en la construcción de los modelos de regresión lineal. Las funciones utilizadas son: *Max*, *Min*, *Mean*, *STD*, $F(Q_1)$, $F(Q_2)$, $F(Q_3)$ y $F(Q_4)$. Estas tablas entregan las estadísticas como la media, la mediana, la desviación estándar, cantidad de valores por bajo 0.15 y cantidad de valores por sobre 0.85, 0.9 y 0.95, consideradas útiles con respecto al objetivo de esta memoria.

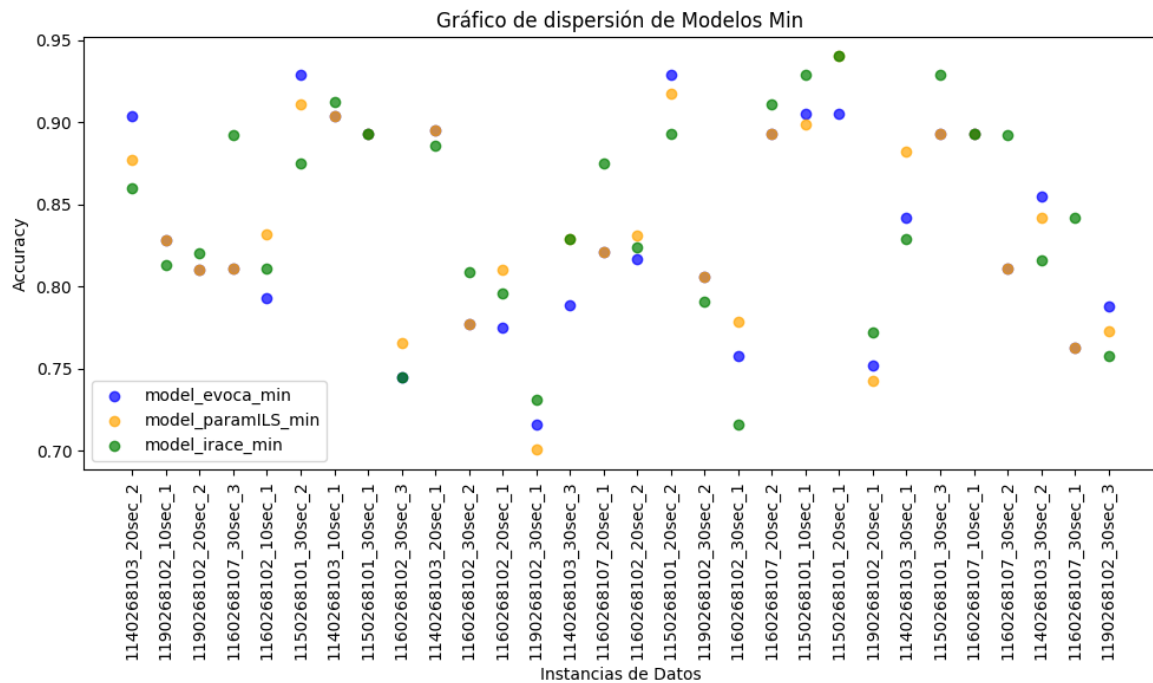


Figura 11: Gráfico de Dispersión de los Modelos *Min* para Evoca, ParamILS e Irace.

Fuente: Elaboración propia

Antes de elegir un tipo de modelo por sobre otro, es importante aclarar que los modelos más útiles en este trabajo son los que demuestren cumplir con los siguientes criterios:

- Valores altos de *exactitud* (cercanos a 1).
- Poca cantidad de resultados con valor NaN o de baja calidad, producidos por algún error dentro de las librerías, normalmente por falta de memoria, utilizadas en el **TM-MA** para ciertas instancias con ciertas configuraciones.
- Consistencia en obtener buenos valores de *exactitud*.

Los criterios anteriores se definen así con el objetivo de entregar un tipo modelo de regresión lineal capaz de mostrar una solidez de resultados frente a nuevas situaciones de datos GPS, como disminuir la cantidad de NaN, presentar buenos valores e intentar competir contra valores de *buffer* y *tol* definidos en la literatura.

Estadística	Min	Max	Mean	STD
Cantidad de Resultados	90	90	90	90
Media	0.837	0.724	0.720	0.810
Mediana	0.829	0.818	0.825	0.829
Desviación Estándar	0.290	0.191	0.298	0.158
Cantidad de NaN	0	0	6	1
Cantidad de Valores Menores que 0.15	0	12	12	3
Cantidad de Valores Mayores que 0.85	38	36	34	38
Cantidad de Valores Mayores que 0.9	15	16	16	21
Cantidad de Valores Mayores que 0.95	0	1	1	2

Tabla 4: Estadísticas de Modelos según Funciones .

Estadística	$F(Q_1)$	$F(Q_2)$	$F(Q_3)$	$F(Q_4)$
Cantidad de Resultados	90	90	90	90
Media	0.840	0.784	0.827	0.843
Mediana	0.830	0.828	0.829	0.843
Desviación Estándar	0.217	0.106	0.298	0.061
Cantidad de NaN	1	1	2	3
Cantidad de Valores Menores que 0.15	0	6	1	0
Cantidad de Valores Mayores que 0.85	41	37	39	43
Cantidad de Valores Mayores que 0.9	14	18	19	19
Cantidad de Valores Mayores que 0.95	0	1	1	1

Tabla 5: Estadísticas de Modelos según Funciones .

Con lo anterior y dadas las estadísticas presentadas en las Tablas 4 y 5, se pueden hacer diferentes elecciones de modelos dependiendo del enfoque que se utilice, si se busca disminuir

en mayor medida la cantidad de NaN y bajos valores de *exactitud* utilizar modelos compuestos con las funciones *Min* o $F(Q_1)$ pueden resultar favorables. A pesar de lo anterior, son las opciones que cuentan con la menor cantidad de altos valores (valores sobre 0.9).

Otro enfoque es buscar siempre la mayor cantidad de valores altos a consecuencia de obtener una cantidad de NaN o valores aún más bajos. Con este enfoque se puede considerar la mayoría de los modelos según sus funciones, pero en especial el modelo construido con *STD* en su proceso de entrenamiento, este tipo de modelos cuentan con la mayor cantidad de valores sobre 0.9 y 0.95, todo lo anterior a costa de 4 valores desfavorables dentro de los experimentos realizados.

Por último, hay dos tipos de modelos que a pesar de no destacar en ninguno de los indicadores anteriores, también cuentan con estadísticas a considerar, primero están los modelos formados por $F(Q_3)$, los cuales presentaron resultados similares a los modelos *STD* mejorando en el apartado de menor cantidad de valores no deseados. Su único defecto es contar con una alta desviación estándar, lo cual puede llegar a generar resultados de calidad más baja de lo esperado. El segundo tipo de modelos son los formados por $F(Q_4)$, los cuales al igual que *STD* muestran una alta cantidad de valores altos, además cuenta con la media y mediana más altas de estos experimentos, lo anterior sumado a contar con la desviación estándar más baja de todas, pueden ser modelos capaces de entregar resultados robustos por sobre el resto.

Esta diferencia puede deberse a la diferencia del cálculo que utilizan este tipos de funciones, la función *STD* es menos sensible a valores atípicos que pueden llegar a encontrarse dentro de datos desconocidos, a diferencia de cómo pueden llegar a ser *Min*, *Max* y *Mean*, los cuales son más sensibles.

Como conclusión de estos resultados de modelos según sus funciones, las funciones *STD* y $F(Q_4)$ se pueden alinear en mayor medida con los objetivos de esta memoria, tomándose en cuenta en conjunto con los resultados de los modelos según el algoritmo sintonizador utilizado en el proceso de ajuste de parámetros.

4.4.2. Resultados Según Sintonizador

Dado que se utilizaron tres algoritmos sintonizadores para ajustar los parámetros del **TMMA**, cuyos resultados se utilizaron como $\vec{y}_{param,sint}$ en la ecuación 4.3, los resultados de esta etapa de experimentación fueron agrupados según dichos algoritmos.

La Tabla 6 muestra las estadísticas que se obtuvieron de todos los modelos de regresión lineal cuyo entrenamiento está relacionado indirectamente con un algoritmo sintonizador.

Al igual que en la sección 4.4.1, se puede contar con diferentes tipos de enfoques para seleccionar los mejores tipos de modelos, dependiendo de sus altos resultados, baja cantidad de valores no deseados o entrega de valores consistente pero sin destacar en ninguno de los

Estadística	ParamILS	Evoca	Irace
Cantidad de Resultados	240	240	240
Media	0.835	0.792	0.770
Mediana	0.857	0.857	0.825
Desviación Estándar	0.119	0.191	0.231
Cantidad de NaN	5	2	7
Cantidad de Valores Menores que 0.15	4	12	18
Cantidad de Valores Mayores que 0.85	120	90	96
Cantidad de Valores Mayores que 0.9	53	51	33
Cantidad de Valores Mayores que 0.95	7	0	0

Tabla 6: Estadísticas de Modelos según Sintonizador.

puntos anteriores. Dado lo anterior, los tipos de modelos que mejor cumplen estos objetivos son los que utilizaron ParamILS en el proceso de sintonización de parámetros.

Los modelos de ParamILs, destacan por tener el menor número de valores de *exactitud* no deseados en comparación de los otros tipos de modelos. Además, cuentan con las soluciones de mejor calidad, siendo los únicos tipos de modelos que entregaron valores por sobre 0.95 a diferencia del resto. Por último, los resultados de ParamILS mostraron la media y mediana más altas, que acompañadas por la menor desviación estándar, son un indicio de poseer la capacidad de entregar altos valores constantes frente a datos desconocidos.

Los modelos de Evoca e Irace se quedan por detrás de los modelos de ParamILS. En especial por sus altos números de valores no esperados en el caso de Irace. Por otro lado Evoca no logra destacar dada su falta de valores aún mayores, además muestra una poca consistencia al contar con una desviación estándar, aún así cuenta con un alto número de buenos resultados pero sin lograr anteponerse a otros modelos.

4.4.3. Resultados Modelos ParamILS

Dado los resultados de los análisis, se decide interiorizar en los resultados que obtuvieron los modelos de ParamILS, para ello en las Tablas 7 y 8 se muestran las estadísticas que se obtuvieron al ejecutar el **TMMA** con parámetros *buffer* y *tol* obtenidos a partir de modelos entrenados con las respectivas funciones y con valores esperados que se consiguieron a través de la sintonización de parámetros utilizando el algoritmo ParamILS.

Estas tablas muestran los resultados únicamente del modelo correspondiente, es decir, ya no es un compuesto de estadísticas de diferentes modelos. Así, es posible visualizar más claramente el comportamiento y resultados posibles que pueden llegar a obtener frente a nuevas instancias de datos.

Si se considera el análisis previo en la sección 4.4.1, al prestar atención a los modelos que

Estadística	Min	Max	Mean	STD
Cantidad de Resultados	30	30	30	30
Media	0.837	0.850	0.861	0.765
Mediana	0.83	0.864	0.878	0.83
Desviación Estándar	0.058	0.061	0.059	0.244
Cantidad de NaN	0	0	0	1
Cantidad de Valores Menores que 0.15	0	0	0	3
Cantidad de Valores Mayores que 0.85	12	16	18	12
Cantidad de Valores Mayores que 0.9	4	6	9	8
Cantidad de Valores Mayores que 0.95	0	1	1	2

Tabla 7: Estadísticas de Modelos de ParamILS según Funciones .

Estadística	$F(Q_1)$	$F(Q_2)$	$F(Q_3)$	$F(Q_4)$
Cantidad de Resultados	30	30	30	30
Media	0.851	0.846	0.812	0.852
Mediana	0.868	0.872	0.839	0.862
Desviación Estándar	0.057	0.066	0.160	0.062
Cantidad de NaN	1	1	1	1
Cantidad de Valores Menores que 0.15	0	0	1	0
Cantidad de Valores Mayores que 0.85	16	15	14	17
Cantidad de Valores Mayores que 0.9	4	7	7	8
Cantidad de Valores Mayores que 0.95	0	1	1	1

Tabla 8: Estadísticas de Modelos de ParamILS según Funciones .

utilizan STD o $F(Q_4)$, se observa que ambos presentan valores no deseados, además de tener otros resultados comparables a otros modelos de ParamILS en cuanto al valor de *exactitud*, dando pie a otro tipos de modelos a considerar como la solución objetivo. Así, a continuación sea comparan los modelos a través de los enfoques, mejores resultados, menor número de resultados no deseables y robustez en resultados aceptables.

En base al primer enfoque, tanto el modelo de STD , $Mean$ y $F(Q_4)$ presentan el mayor número de resultados de alta calidad (por sobre 0.9). En cambio para el segundo enfoque, los modelos de Min , Max y $Mean$ no presentan valores no deseables, aunque en general los modelos no tuvieron demasiados de estos valores a excepción de STD . Finalmente, en un enfoque de robustez de resultados, se tiene que todos los modelos no poseen una alta variabilidad en sus resultados que según sus estadísticas son altos, pero un modelo destaca sobre el resto en cuanto a media y mediana, siendo este el modelo $Mean$ el cual presenta los resultados más consistentes.

En el análisis anterior, se selecciona el modelo $Mean$ como el que cumple todas las exigencias de calidad de resultados, lo que lo convierte en el modelo candidato a ser añadido a la versión adaptativa del **TMMA**.

4.5. Resultados Parámetros Genéricos y Modelo adaptativo

Para finalizar con la validación de la solución de este trabajo de memoria, se presenta una comparación de los resultados del algoritmo **TMMA** con valores de parámetros obtenidos con los modelos de regresión lineal *Mean* versus los resultados de ejecutar el algoritmo **TMMA** usando valores para los parámetros *buffer* y *tol* definidos según la literatura.

Las instancias que se utilizan para esta comparación corresponden a las utilizadas en los experimentos de la sección 4.4. Así, se cuenta con un total de 30 instancias de datos las cuales entregarán un total de 30 resultados.

Los valores de *buffer* y *tol* genéricos son extraídos de [Leon *et al.*, 2020] y [Blazquez y Vonderohe, 2009], los cuales coinciden en utilizar *buffer* con un valor de 60 y 25 para el parámetro *tol* en los datos provenientes de la ciudad de Portage, Wisconsin, Estados Unidos.

La Tabla 14 muestra los resultados obtenidos a partir de esta comparación.

En estos resultados, se observa que, en general, el **TMMA**, utilizando los valores de *buffer* y *tol* obtenidos de la literatura, obtuvo mejores resultados en 17 de los 30 casos. En cambio, los resultados basados en la predicción de parámetros de los modelos ParamILS/*Mean* fueron superiores en solo ocho instancias. Ambos métodos lograron el mismo resultado en cinco de las 30 instancias de ejecución. Lo anterior podría indicar que las predicciones de los modelos de ParamILS/*Mean* no son capaces de obtener desempeños comparables para conjuntos de datos ya conocidos, no obstante, el comportamiento de los resultados mostrados en la Figura 12, muestran que ambos tienden a seguir los mismos valores de *exactitud*. Además se puede ver de forma gráfica la diferencia entre estos resultados, pudiendo compararse en detalle de acuerdo a la instancia de datos.

En la Tabla 9 se presenta la diferencia entre de resultados obtenidos del **TMMA** adaptativo con modelos ParamILS/*Mean* y los resultados obtenidos del **TMMA** con valores genéricos. Los resultados positivos muestran los casos en que la propuesta de algoritmo adaptativo logra obtener un valor de *exactitud* favorable con respecto al **TMMA** con valores genéricos. Los valores negativos representan una desmejora de calidad con respecto a los resultados de utilizar configuraciones encontradas en la literatura.

Así, el modelo de regresión lineal que se añade al **TMMA** puede ser de utilidad, comparado con la ejecución del **TMMA** con una configuración genérica previamente obtenida de la literatura dado que no requiere de un nuevo proceso de determinación de valores.

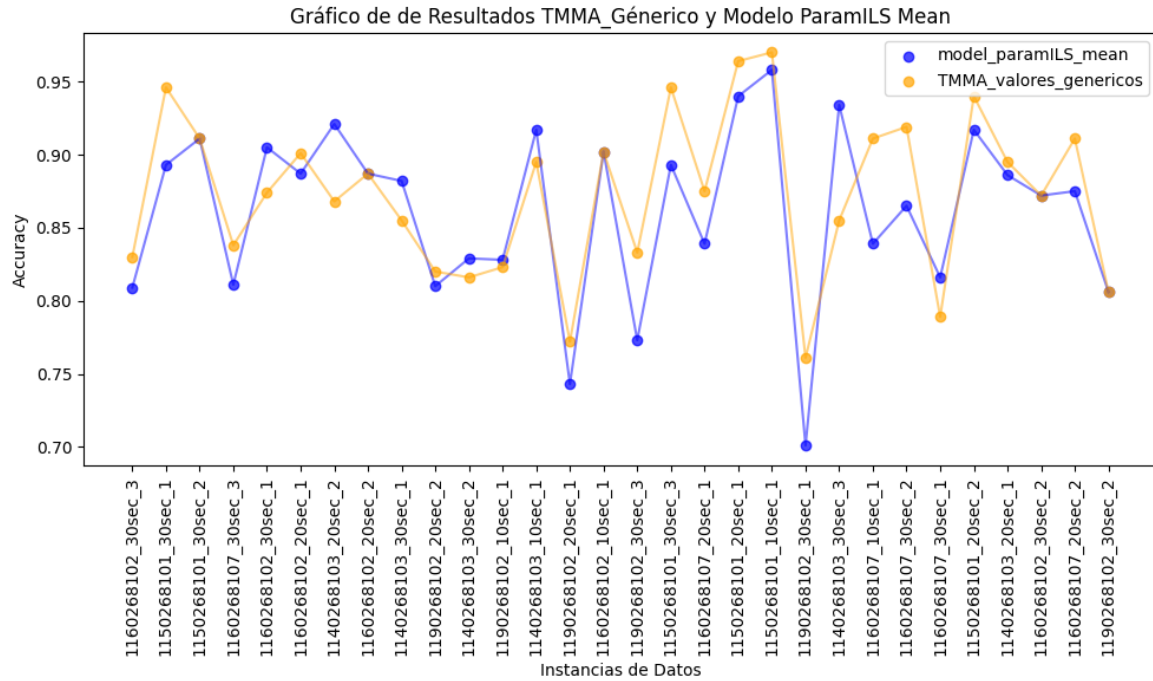


Figura 12: Gráfico de Resultados de **TMMA** Adaptativo con Modelos ParamILS/*Mean* y TMMA con valores genéricos de la literatura.

Fuente: Elaboración propia

Instancia de Datos	Diferencia	Instancia de Datos	Diferencia
1140268103_10sec_1	+0.022	1160268102_30sec_2	0.000
1140268103_20sec_1	-0.009	1160268102_30sec_3	-0.021
1140268103_20sec_2	+0.053	1160268107_10sec_1	-0.072
1140268103_30sec_1	+0.027	1160268107_20sec_1	-0.036
1140268103_30sec_2	+0.013	1160268107_20sec_2	-0.036
1140268103_30sec_3	+0.079	1160268107_30sec_1	+0.027
1150268101_10sec_1	-0.012	1160268107_30sec_2	-0.054
1150268101_20sec_1	-0.024	1160268107_30sec_3	-0.027
1150268101_20sec_2	-0.023	1190268102_10sec_1	+0.005
1150268101_30sec_1	-0.053	1190268102_20sec_1	-0.029
1150268101_30sec_2	0.000	1190268102_20sec_2	-0.010
1150268101_30sec_3	-0.053	1190268102_30sec_1	-0.060
1160268102_10sec_1	0.000	1190268102_30sec_2	0.000
1160268102_20sec_1	-0.014	1190268102_30sec_3	-0.060
1160268102_20sec_2	0.000	1160268102_30sec_1	+0.031

Tabla 9: Diferencias de Resultados del **TMMA** Adaptativo con Modelos ParamILS/*Mean* y **TMMA** con valores genéricos.

4.6. Conclusiones del Capítulo

En este capítulo se presenta la evaluación de la propuesta de solución. Se evalúa un conjunto de ocho modelos con la finalidad de obtener una herramienta capaz de predecir parámetros y obtener resultados ante nuevas instancias de pruebas. Finalmente, se realizó una comparación entre los resultados del **TMMA** obtenidos mediante una configuración de parámetros obtenida de la literatura para un tipo dado de problemas y los resultados generados por una configuración de parámetros adaptativos ejecutada por el algoritmo **TMMA**.

A partir de esto se puede afirmar que los modelos de regresión lineal, cuyos valores de entrenamiento fueron calculados a partir de las columnas de los datos GPS utilizando la función *Mean* y cuyos valores objetivo fueron obtenidos mediante la sintonización de parámetros utilizando el algoritmo ParamILS en los parámetros *buffer* y *tol*, no logran superar en resultados a los parámetros genéricos para la ciudad de Portage. Sin embargo, no se consideran resultados desfavorables, ya que demostraron un comportamiento similar e incluso superaron en algunas ocasiones a los resultados de la versión genérica. Esto sugiere la posibilidad de aumentar la complejidad de los modelos para mejorar sus resultados. Otra opción es la capacidad de utilizar estos modelos ante instancias desconocidas en la literatura como un primer acercamiento, con la capacidad de obtener resultados de buena calidad sin previo conocimiento de la situación.

En el siguiente capítulo se presenta las conclusiones generales de esta memoria.

CAPÍTULO 5

CONCLUSIONES

5.1. Conclusiones Generales

En esta memoria se diseñó, implementó y evaluó el rendimiento de un algoritmo adaptativo con el objetivo de mejorar la solución existente en la literatura para el problema de resolución de ambigüedad espacial. Dada la dificultad de obtener configuraciones para los valores de los parámetros *buffer* y *tol* que proporcionen resultados de alta calidad en diversas situaciones, se propone una metodología adaptativa respecto a las características del problema a resolver.

Con el objetivo anterior, se propuso una metodología de trabajo que, en primera instancia, utiliza las configuraciones de parámetros obtenidas de resultados de diversos algoritmos sincronizadores, herramientas algorítmicas que buscan mejorar el rendimiento de modelos en base a los ajustes de parámetros de éstos. Para, posteriormente, utilizar estas configuraciones como los valores objetivos de una serie de modelos de regresión lineal que buscan predecir nuevas configuraciones adaptadas a casos de prueba desconocidos.

Se evaluaron un total de veinticuatro pares de modelos de regresión lineal para los parámetros *buffer* y *tol*, cuyas predicciones de parámetros se utilizaron en el algoritmo **TMMA**. Estas predicciones se compararon entre sí para seleccionar la mejor combinación para ser incorporada al algoritmo adaptativo. A partir de estos resultados se observa que los modelos de ParamILS/*Mean* obtuvieron los mejores resultados.

El algoritmo adaptativo obtenido fue comparado con la literatura concluyendo que no supera los resultados obtenidos con configuraciones basadas en heurísticas más avanzadas y previamente estudiadas. No obstante, se propone el uso de esta metodología como enfoque preliminar para nuevas situaciones o instancias en las que los estudios avanzados aún estén en sus primeras etapas, dado que puede proporcionar resultados de calidad de manera consistente.

5.2. Cumplimiento de objetivos

Se puede medir el resultado de esta memoria en base de los objetivos específicos planteados en la sección 1.1. A continuación se presentan el cumplimiento de los objetivos específicos.

5.2.1. Objetivos Específicos

El cumplimiento de estos objetivos demuestra el trabajo y avance en pos de entregar la solución planteada como objetivo general.

- **Presentar un algoritmo adaptativo para la selección de parámetros de TMMAs utilizando e integrando conocimientos de algoritmos sintonizadores.**

En la sección 4.1 se realizaron experimentos utilizando algoritmos sintonizados para obtener configuraciones de parámetros capaces de entregar resultados de calidad en las instancias de entrenamiento seleccionadas. Posteriormente en la sección 4.3 se realizan experimentos con modelos de regresión lineal, cuya construcción utiliza directamente los resultados de las sintonizaciones de parámetros previas. Finalmente se hace uso de estos modelos para la selección de parámetros de **TMMA** adaptando sus configuraciones en instancias de prueba desconocidas. Dado lo anterior, se considera que se ha cumplido el objetivo.

- **Analizar y evaluar el rendimiento de los parámetros obtenidos a través del algoritmo a presentar en diferentes casos de prueba.**

En la sección 4.3 se presentan diversos experimentos de modelos de regresión lineal, donde en conjunto a una versión del **TMMA** se consiguen adaptar los parámetros *buffer* y *tol* frente a un total de 30 instancias de pruebas. Posteriormente se evalúa el rendimiento de éstos y se realizan comparaciones de desempeño debidas. Dado lo anterior, se considera que se ha cumplido el objetivo.

- **Realizar una comparación del rendimiento de TMMAs con parámetros obtenidos con el algoritmo adaptativo y el TMMAs con parámetros genéricos con diferentes casos de prueba.**

En la sección 4.5, se realiza una comparación de los resultados obtenidos por el **TMMA** en 30 instancias de prueba. Estos resultados se obtuvieron utilizando dos enfoques diferentes: uno empleando un modelo de regresión lineal con parámetros ajustados por ParamILS y componentes calculadas mediante la función *Mean*, y otro utilizando una configuración de parámetros genérica obtenida de la literatura para datos de Portage. Dado lo anterior, se considera que se ha cumplido el objetivo.

Considerando los objetivos establecidos anteriormente como cumplidos, el objetivo general de proporcionar un algoritmo adaptativo capaz de mejorar los resultados existentes y adaptarse a diversas situaciones no se ha alcanzado por completo. Sin embargo, se ha desarrollado una metodología que permite predecir estos parámetros con buena calidad en nuevas situaciones, sin necesidad de conocimientos previos en el tema.

5.3. Trabajo Futuro

A continuación se presentan recomendaciones a considerar en trabajos a futuro que podrían ser útiles tanto conceptualmente respecto al problema a resolver como también respecto a la forma de trabajo.

1. Obtener la licencia de ArcGIS y ArcMAP para un sistema sobre el cual el usuario tenga control total. Esto no solo mejoraría los tiempos de experimentación debido a las mejoras en el hardware, sino que también resolvería las dificultades encontradas en este estudio al depender de un sistema externo para llevar a cabo las ejecuciones del **TMMA**, lo que hubiese mejorado significativamente los resultados obtenidos.
2. En caso de contar con un mayor número de instancias de datos GPS al inicio del trabajo, se pueden obtener más configuraciones óptimas de los algoritmos sintonizadores, lo cual debería significar una mejora en cuanto a la predicción de los modelos de regresión.
3. En caso de contar con casos de mediciones GPS provenientes de diferentes localidades, se puede generar una mejora en este estudio dotándolo de la capacidad de adaptarse a nuevas zonas y aumentando su utilidad para nuevos usuarios.
4. Explorar funciones o métodos más complejos para predecir configuraciones de parámetros podría mejorar la calidad de las soluciones obtenidas, proporcionando resultados más refinados y precisos.

ANEXOS

Conjunto de herramientas

Para el correcto funcionamiento de esta memoria se hace uso de los software listados a continuación:

ArcGIS y ArcMAP

Estas herramientas son esenciales y necesarios para el funcionamiento del **TMMA** y además permiten el uso de software para visualizar los datos o de información con las cuales obtener una congruencia en los mapas a trabajar dentro de las diferentes instancias de datos.

Instancias de Datos de Prueba

Conjunto de Datos 1	Conjunto de Datos 2	Conjunto de Datos 3
1140268103_10sec_1	1150268101_10sec_1	1160268102_10sec_1
1140268103_20sec_1	1150268101_20sec_1	1160268102_20sec_1
1140268103_20sec_2	1150268101_20sec_2	1160268102_20sec_2
1140268103_30sec_1	1150268101_30sec_1	1160268102_30sec_1
1140268103_30sec_2	1150268101_30sec_2	1160268102_30sec_2
1140268103_30sec_3	1150268101_30sec_3	1160268102_30sec_3

Tabla 10: Conjuntos de Instancias 1, 2 y 3 de Datos de Prueba.

Instancias de Datos 4	Instancias de Datos 5
1190268102_10sec_1	1160268107_10sec_1
1190268102_20sec_1	1160268107_20sec_1
1190268102_20sec_2	1160268107_20sec_2
1190268102_30sec_1	1160268107_30sec_1
1190268102_30sec_2	1160268107_30sec_2
1190268102_30sec_3	1160268107_30sec_3

Tabla 11: Conjuntos de Instancias 4 y 5 de Datos de Prueba.

Gráficos de Dispersión

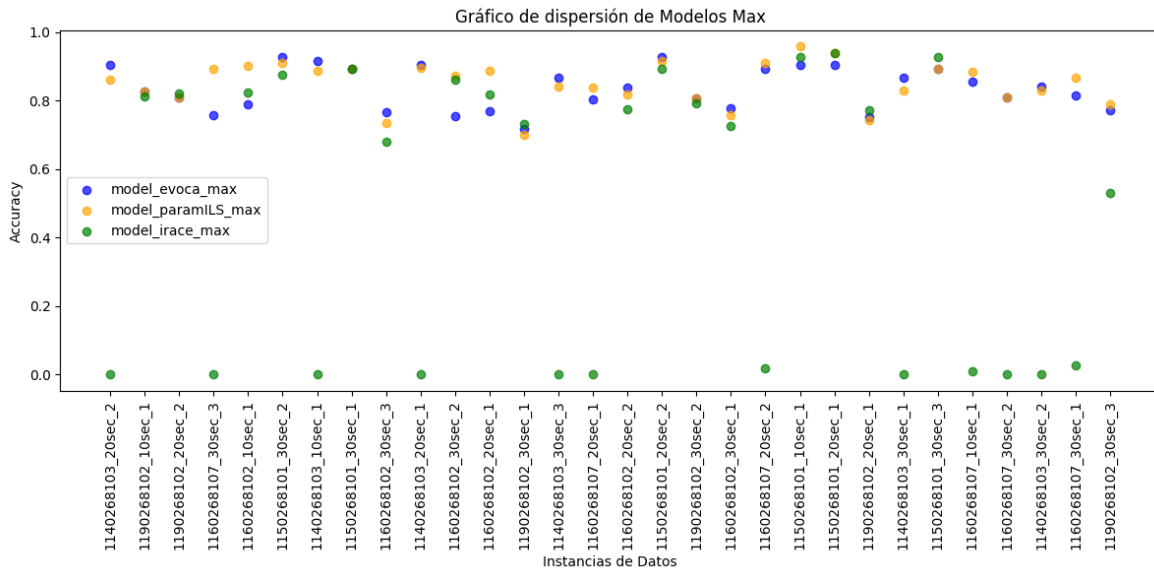


Figura 13: Gráfico de Dispersión de los Modelos Max para Evoca, ParamILS e Irace.
Fuente: Elaboración propia

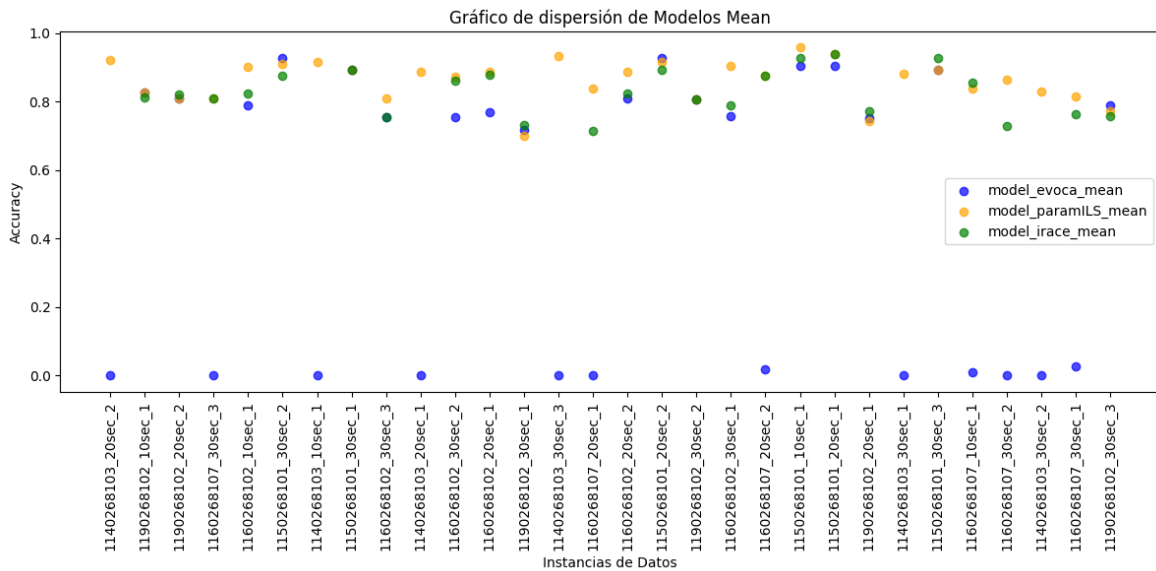


Figura 14: Gráfico de Dispersión de los Modelos Mean para Evoca, ParamILS e Irace.
Fuente: Elaboración propia

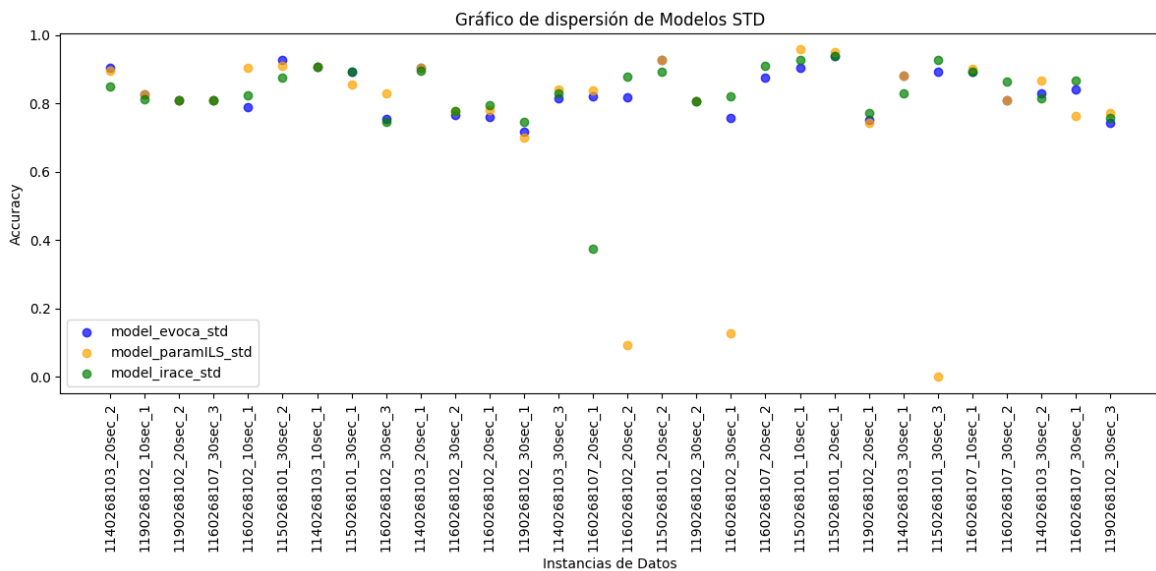


Figura 15: Gráfico de Dispersión de los Modelos STD para Evoca, ParamILS e Irace.
Fuente: Elaboración propia

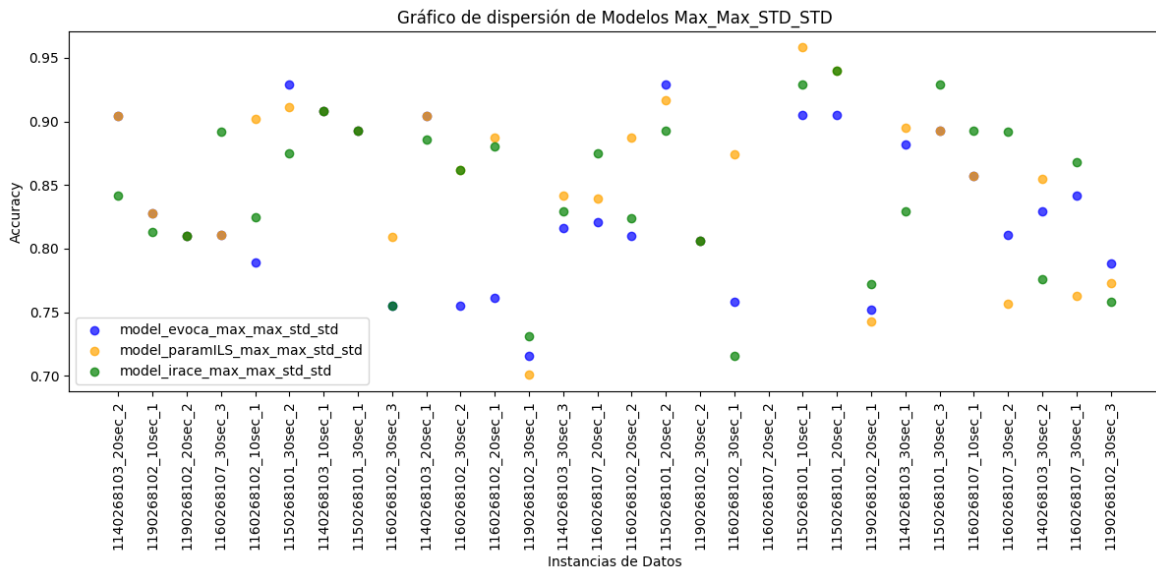


Figura 16: Gráfico de Dispersión de los Modelos Max_Max_STD_STD para Evoca, ParamILS e Irace.

Fuente: Elaboración propia

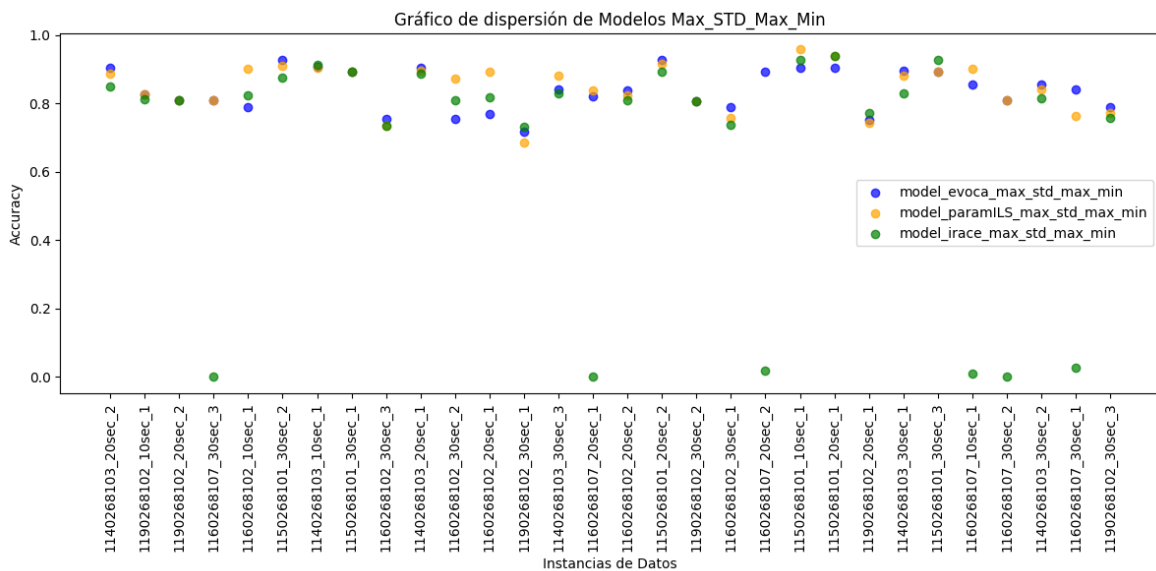


Figura 17: Gráfico de Dispersión de los Modelos Max_STD_Max_Min para Evoca, ParamILS e Irace.

Fuente: Elaboración propia

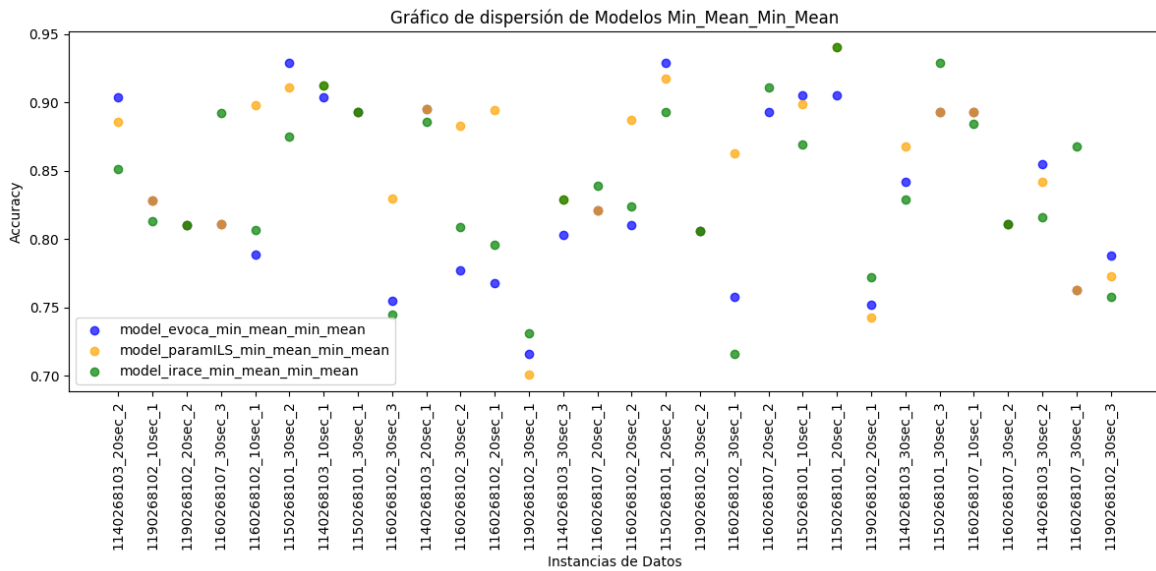


Figura 18: Gráfico de Dispersión de los Modelos Min_Mean_Min_Mean para Evoca, ParamILS e Irace.

Fuente: Elaboración propia

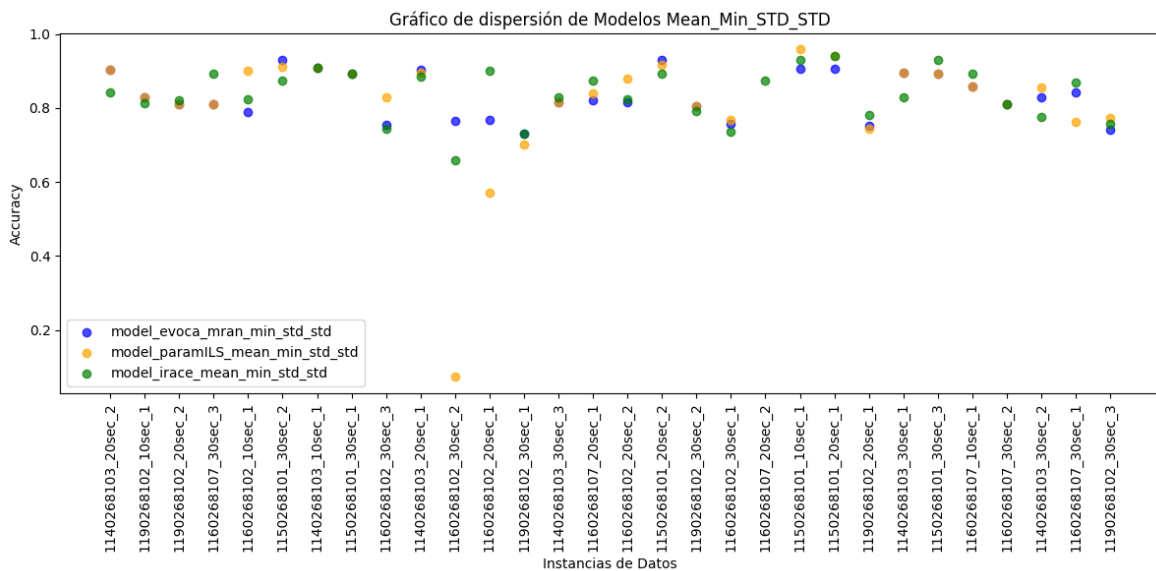


Figura 19: Gráfico de Dispersión de los Modelos Mean_Min_STD_STD para Evoca, ParamILS e Irace.

Fuente: Elaboración propia

Coeficientes de Modelos de Regresión Lineal

Modelos	Coef1	Coef2	Coef3	Coef4	Inter
evoca_max	-0.0114	-0.0011	0.344	-0.0001	2340
evoca_max_max_std_std	0.00292	0.00142	0.890	-0.00972	-804
evoca_max_std_max_min	-0.00490	-0.00096	0.202	-8.22e-05	911
evoca_min	-0.00046	2.62e-05	0.0	7.27e-05	97.9
evoca_min_mean_min_mean	-0.00068	-0.00029	0.0	9.06e-05	200
evoca_mean	-0.0218	-0.0144	-1.17	0.00088	6800
evoca_mean_min_std_std	-0.00202	0.00032	2.64	0.0139	284
evoca_std	0.00335	-0.00140	2.18	0.00777	-5.61
irace_max	0.0	0.0	0.0	0.0	10.0
irace_max_max_std_std	0.0	0.0	0.0	0.0	10.0
irace_max_std_max_min	0.0	0.0	0.0	0.0	10.0
irace_min	0.0	0.0	0.0	0.0	10.0
irace_min_mean_min_mean	0.0	0.0	0.0	0.0	10.0
irace_mean	0.0	0.0	0.0	0.0	10.0
irace_mean_min_std_std	0.0	0.0	0.0	0.0	10.0
irace_std	0.0	0.0	0.0	0.0	10.0
paramILS_max	0.0223	0.00332	-0.541	0.00021	-4690
paramILS_max_max_std_std	0.00186	-0.00058	-1.99	0.0113	-170
paramILS_max_std_max_min	0.00310	0.00264	-0.131	-4.44e-05	-537
paramILS_min	-0.00030	-0.00069	0.0	-0.00011	217
paramILS_min_mean_min_mean	-0.00042	-0.00092	0.0	-0.00017	294
paramILS_mean	0.0170	0.0126	-0.231	-0.00085	-5480
paramILS_mean_min_std_std	-0.00047	-0.00112	-1.39	-0.00190	345
paramILS_std	0.00110	-0.0242	-6.20	0.154	127

Tabla 12: Coeficientes e Intercepto de Modelos de Predicción del Parámetro *tol*

Modelos	Coef1	Coef2	Coef3	Coef4	Inter
evoca_max	-0.164	0.00131	5.31	-0.00111	29400
evoca_max_max_std_std	-0.00129	0.0370	29.5	-0.0437	-7590
evoca_max_std_max_min	-0.172	0.00046	5.47	-0.00122	31200
evoca_min	-0.022	-0.00351	0.0	0.00374	4530
evoca_min_mean_min_mean	-0.028	-0.0122	0.0	0.00381	7320
evoca_mean	-0.51	-0.312	-41.5	0.0214	154000
evoca_mean_min_std_std	-0.020	0.0334	41.6	0.665	-3530
evoca_std	0.025	0.594	157	-3.39	-2560
irace_max	0.538	0.0923	-15.2	0.00902	-116000
irace_max_max_std_std	-0.162	-0.0255	-23.6	0.539	34700
irace_max_std_max_min	0.00507	0.0750	-3.82	0.00197	-1020
irace_min	-0.0115	-0.00911	0.0	0.00069	3830
irace_min_mean_min_mean	-0.00657	-0.00283	0.0	-0.000512	1850
irace_mean	0.594	0.447	1.65	-0.024	-194000
irace_mean_min_std_std	0.0826	0.0300	-102	0.243	-19100
irace_std	-0.148	0.915	85.2	-5.14	-1890
paramILS_max	-0.0217	0.00179	-4.21	0.00427	3800
paramILS_max_max_std_std	-0.286	-0.0355	12.0	0.275	59200
paramILS_max_std_max_min	-0.0305	0.00347	-3.99	0.00417	5750
paramILS_min	0.00875	0.0201	0.0	0.00191	-5230
paramILS_min_mean_min_mean	0.0271	0.0481	0.0	0.00267	-14300
paramILS_mean	0.560	0.386	41.5	-0.017	-177000
paramILS_mean_min_std_std	0.141	0.0615	-123	-0.0632	-35000
paramILS_std	-0.256	1.77	238	-10.6	-4290

Tabla 13: Coeficientes e Intercepto de Modelos de Predicción del Parámetro *buffer*

Resultados de Comparación

Instancia de Datos	Modelo ParamILS/ <i>Mean</i>	TMMA Genérico
1140268103_10sec_1	0.917	0.895
1140268103_20sec_1	0.886	0.895
1140268103_20sec_2	0.921	0.868
1140268103_30sec_1	0.882	0.855
1140268103_30sec_2	0.829	0.816
1140268103_30sec_3	0.934	0.855
1150268101_10sec_1	0.958	0.97
1150268101_20sec_1	0.94	0.964
1150268101_20sec_2	0.917	0.94
1150268101_30sec_1	0.893	0.946
1150268101_30sec_2	0.911	0.911
1150268101_30sec_3	0.893	0.946
1160268102_10sec_1	0.902	0.902
1160268102_20sec_1	0.887	0.901
1160268102_20sec_2	0.887	0.887
1160268102_30sec_1	0.905	0.874
1160268102_30sec_2	0.872	0.872
1160268102_30sec_3	0.809	0.83
1160268107_10sec_1	0.839	0.911
1160268107_20sec_1	0.839	0.875
1160268107_20sec_2	0.875	0.911
1160268107_30sec_1	0.816	0.789
1160268107_30sec_2	0.865	0.919
1160268107_30sec_3	0.811	0.838
1190268102_10sec_1	0.828	0.823
1190268102_20sec_1	0.743	0.772
1190268102_20sec_2	0.81	0.82
1190268102_30sec_1	0.701	0.761
1190268102_30sec_2	0.806	0.806
1190268102_30sec_3	0.773	0.833

Tabla 14: Resultados de **TMMA** Adaptativo con Modelos ParamILS/*Mean* y TMMA Genérico.

REFERENCIAS BIBLIOGRÁFICAS

- [Bang *et al.*, 2016] Bang, Y., Kim, J., y Yu, K. (2016). An improved map-matching technique based on the fr chet distance approach for pedestrian navigation services. *Sensors*, 16(10).
- [Birattari *et al.*, 2002] Birattari, M., St tztle, T., Paquete, L., y Varrentrapp, K. (2002). A racing algorithm for configuring metaheuristics. pp. 11–18.
- [Blazquez *et al.*, 2022] Blazquez, C., Le n, R., P rez, F., y Delgado, L. (2022). Pre-processing techniques for improving a topological map-matching algorithm. En *2022 41st International Conference of the Chilean Computer Science Society (SCCC)*, pp. 1–6.
- [Blazquez *et al.*, 2018] Blazquez, C., Ries, J., Miranda, P. A., y Le n, R. J. (2018). An instance-specific parameter tuning approach using fuzzy logic for a post-processing topological map-matching algorithm. *IEEE Intelligent Transportation Systems Magazine*, 10(4):87–97.
- [Blazquez y Vonderohe, 2009] Blazquez, C. y Vonderohe, A. (2009). Effects of controlling parameters on performance of a decision-rule map-matching algorithm. *Journal of Transportation Engineering-asce - J TRANSP ENG-ASCE*, 135.
- [Chao *et al.*, 2020] Chao, P., Xu, Y., Hua, W., y Zhou, X. (2020). A survey on map-matching algorithms. En Borovica-Gajic, R., Qi, J., y Wang, W., editores, *Databases Theory and Applications*, pp. 121–133, Cham. Springer International Publishing.
- [Efrat *et al.*, 2002] Efrat, A., Guibas, L., Har-Peled, S., Mitchell, J., y Murali (2002). New similarity measures between polylines with applications to morphing and polygon sweeping. *Discrete and Computational Geometry*, 28:535–569.
- [Esri, 2012] Esri (2012). Geographic information systems (gis).
- [Hutter *et al.*, 2007] Hutter, F., Hoos, H., y St tztle, T. (2007). Automatic algorithm configuration based on local search.
- [Leon *et al.*, 2020] Leon, R., Blazquez, C., y Depassier, V. (2020). Improvements of a topological map-matching algorithm in post-processing mode. En *2020 39th International Conference of the Chilean Computer Science Society, SCCC 2020, Proceedings - International Conference of the Chilean Computer Science Society, SCCC, United States. IEEE Computer Society*.
- [L pez-Ib  ez *et al.*, 2016a] L pez-Ib  ez, M., Dubois-Lacoste, J., P rez C ceres, L., Birattari, M., y St tztle, T. (2016a). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.
- [L pez-Ib  ez *et al.*, 2016b] L pez-Ib  ez, M., Dubois-Lacoste, J., P rez C ceres, L., Birattari, M., y St tztle, T. (2016b). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

- [Menezes *et al.*, 2020] Menezes, M. C., Muñoz, M. E. S., Freitas, E. P., Cheng, S., Walther, T., Neto, A. A., Ribeiro, P. R. A., y Oliveira, A. C. M. (2020). Automatic tuning of ratslam's parameters by irace and iterative closest point. En IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society, pp. 562–568.
- [Montero *et al.*, 2014] Montero, E., Riff, M.-C., y Neveu, B. (2014). A beginner's guide to tuning methods. Applied Soft Computing, 17:39–51.
- [Nannen y Eiben, 2007] Nannen, V. y Eiben, A. (2007). Relevance estimation and value calibration of evolutionary algorithm parameters. pp. 103 – 110.
- [Prestes *et al.*, 2018] Prestes, L., Delgado, M. R., Lüders, R., Gonçalves, R., y Almeida, C. P. (2018). Boosting the performance of moea/d-dra with a multi-objective hyper-heuristic based on irace and ucb method for heuristic selection. En 2018 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8.
- [Riff y Montero, 2013] Riff, M. y Montero, E. (2013). A new algorithm for reducing metaheuristic design effort. En 2013 IEEE Congress on Evolutionary Computation, pp. 3283–3290.
- [Rojas-Morales y Riff, 2023] Rojas-Morales, N. y Riff, M. (2023). Opposite scoring: focusing the tuning process of evolutionary calibrator. Neural Computing and Applications, 35.
- [Szczepanski *et al.*, 2020] Szczepanski, N., Mousin, L., Veerapen, N., y Jourdan, L. (2020). Automatic configuration of multi-thread local search: Preliminary results on bi-objective tsp. En 2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI), pp. 1241–1248.