ALGORITMO MEMÉTICO MULTI-OBJETIVO PARA CALIBRAR MODELOS DE MICRO-SIMULACIÓN DE FLUJO DE TRÁFICO VEHICULAR CORSIM

Cristian Camilo Erazo Agredo
Julio César Luna Ortega

Director: PhD. Carlos Alberto Cobos Lozada
Co-Director: PhD. Martha Eliana Mendoza Becerra
Co-director externo: PhD. Alexander Paz (University of Nevada)
Asesores: Carlos Gaviria y Cristian Arteaga (University of Nevada)

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Línea de Investigación de Sistemas Inteligentes
Popayán, 2017

TABLA DE CONTENIDO

1. INTRODUCCIÓN 9
1.1 PLANTEAMIENTO DEL PROBLEMA 9
1.2 JUSTIFICACIÓN 11
1.3 OBJETIVOS 12
1.3.1 OBJETIVO GENERAL 12
1.3.2 OBJETIVOS ESPECÍFICOS 12
1.4 RESULTADOS OBTENIDOS 13
1.5 ESTRUCTURA DE LA MONOGRAFÍA 13
2. CONTEXTO TEÓRICO 15
2.1 INTRODUCCIÓN 15
2.2 SIMULACIÓN Y MICRO-SIMULACIÓN 15
2.3 MODELO DE MICRO-SIMULACIÓN CORSIM 15
2.4 CALIBRACIÓN DE MODELOS DE SIMULACIÓN DE TRÁFICO VEHICULAR 16
2.5 ALGORITMO DE OPTIMIZACIÓN 17
2.6 ALGORITMO MEMÉTICO 18
2.7 ALGORITMO MULTI-OBJETIVO 19
2.7.1 ALGORITMO NSGA-II 20
2.7.2 ALGORITMO MOGBHS 21
2.7.3 ALGORITMO SPEA-2 22
2.8 ALGORITMOS DE BÚSQUEDA LOCAL 22
2.8.1 ALGORITMO DE ASCENSO A LA COLINA 23
2.8.2 ALGORITMO DE RECOCIDO SIMULADO 23
2.8.3 ALGORITMO DE BÚSQUEDA LOCAL ITERADA 24
2.9 ARREGLOS DE COBERTURA 25
3. ESTADO DEL ARTE 26
3.1 INTRODUCCIÓN 26
3.2 CALIBRACIÓN DE MODELOS DE TRÁFICO VEHICULAR 26
3.3 CALIBRACIÓN MEDIANTE ALGORITMOS MULTI-OBJETIVO 29
3.4 OPTIMIZACIÓN USANDO NSGA-II Y MOGBHS 30
4. PROPUESTA 32
4.1 INTRODUCCIÓN 32
4.2 FUNCIONES OBJETIVO Y CRITERIO DE CALIBRACIÓN 32

4.3 REPRESENTACIÓN DE UN INDIVIDUO 33
4.4 DESCRIPCIÓN DE UNA SOLUCIÓN VECINA 35
4.4.1 SOLUCIÓN VECINA GENERADA A PARTIR DEL ALGORITMO NSGA-II 35
4.4.2 SOLUCIÓN VECINA GENERADA A PARTIR DEL ALGORITMO MOGBHS 36
4.4.3 SOLUCIÓN VECINA GENERADA A PARTIR DEL ALGORITMO SPEA-2 36
4.4.4 SOLUCIÓN VECINA GENERADA A PARTIR DE LOS ALGORITMOS DE BÚSQUEDA LOCAL 37
4.5 DESCRIPCIÓN DE UNA MEJOR SOLUCIÓN 37
4.6 ALGORITMO DE BÚSQUEDA LOCAL: ASCENSO A LA COLINA 37
4.7 ALGORITMO DE BÚSQUEDA LOCAL: RECOCIDO SIMULADO 39
4.8 ALGORITMO DE BÚSQUEDA LOCAL: BÚSQUEDA LOCAL ITERADA . 40
4.9 ALGORITMO GENÉTICO BASADO EN ORDENAMIENTO NO DOMINADO-II (NSGA-II) 41
4.10 MEJOR BÚSQUEDA GLOBAL ARMÓNICA MULTI-OBJETIVO (MOGBHS) 43
4.11 ALGORITMO MULTI-OBJETIVO DE FUERZA DE PARETO-2 (SPEA-2) 44
4.12 PROPUESTA MEMÉTICA DE LOS ALGORITMOS MULTI-OBJETIVO 47
4.12.1 PROPUESTA MEMÉTICA DEL ALGORITMO NSGA-II 47
4.12.2 PROPUESTA MEMÉTICA DEL ALGORITMO MOGBHS 48
4.12.3 PROPUESTA MEMÉTICA DEL ALGORITMO SPEA2 48
4.12.4 MÉTODO DE EXPLOTACIÓN 48
4.13 USO DE HILOS 48
4.13.1 HILOS EN LA EVALUACIÓN DE LA APTITUD 49
4.13.2 HILOS EN LOS ALGORITMOS PROPUESTOS 49
5. EXPERIMENTACIÓN 51
5.1 MODELOS DE PRUEBA 51
5.1.1 MODELO DE COMPLEJIDAD BAJA: MCTRANS NETWORK 51
5.1.2 MODELO DE COMPLEJIDAD MEDIA: RENO NETWORK 52
5.1.3 MODELO DE COMPLEJIDAD ALTA: I-75 NETWORK 53
5.2 ALGORITMOS A COMPARAR Y MEDIDAS 54
5.3 AFINAMIENTO DE PARÁMETROS 54

5.4 RESULTADOS OBTENIDOS Y COMPARACIÓN 58
5.4.1 RESULTADOS Y COMPARACIÓN PARA EL MODELO MCTRANS 59
5.4.2 RESULTADOS OBTENIDOS PARA EL MODELO RENO 70
5.4.3 RESULTADOS OBTENIDOS PARA EL MODELO I-75 81
5.5 ANÁLISIS DE RESULTADOS 91
5.6 ANÁLISIS DE SENSIBILIDAD 92
6. CONCLUSIONES Y TRABAJOS FUTUROS 95
6.1 CONCLUSIONES 95
6.2 TRABAJOS FUTUROS 96
7. BIBLIOGRAFÍA 98

ÍNDICE DE FIGURAS

Figura 1: Ejemplo del frente de Pareto. 20
Figura 2: Ejemplo $CA(9;\ 2,4,3)$. 25
Figura 3. Estructura del archivo plano en formato de CORSIM. 34
Figura 4. Ejemplo de la representación de la solución (vector de parámetros). 34
Figura 5: Pseudocódigo algoritmo ascenso a la colina. 38
Figura 6: Pseudocódigo Método Domina A. 39
Figura 7: Pseudocódigo algoritmo recocido simulado. 40
Figura 8: Pseudocódigo algoritmo búsqueda local iterada. 41
Figura 9: Pseudocódigo algoritmo NSGA-II. 42
Figura 10: Pseudocódigo Método Reproducir. 43
Figura 11: Pseudocódigo algoritmo MOGBHS. 44
Figura 12: Pseudocódigo algoritmo SPEA-2. 46
Figura 13: Pseudocódigo Método Selección Entorno. 47
Figura 14: Pseudocódigo Método BusquedaLocal. 48
Figura 15: Imagen generada con el software TRAFVU Viewer a partir del modelo CORSIM de la red McTrans. 52
Figura 16: Autopista Pyramid, Reno, Nevada. Tomado de Google Maps y TRAFVU Viewer. 53
Figura 17: Autopista I-75, Miami, FL. Tomado de Google Maps y TRAFVU Viewer. 54
Figura 18: Gráfica NRMS vs Tiempo en la red McTrans del algoritmo NSGA-II y su propuesta memética que incluye HC. 60
Figura 19: Gráfica NRMS vs Tiempo en la red McTrans del algoritmo NSGA-II y su propuesta memética que incluye SA. 60
Figura 20: Gráfica NRMS vs Tiempo en la red McTrans del algoritmo NSGA-II y su propuesta memética que incluye ILS. 61
Figura 21: Gráfica NRMS vs Tiempo en la red McTrans del algoritmo MOGBHS y su propuesta memética que incluye HC. 62
Figura 22: Gráfica NRMS vs Tiempo en la red McTrans del algoritmo MOGBHS y su propuesta memética que incluye SA. 63
Figura 23: Gráfica NRMS vs Tiempo en la red McTrans del algoritmo MOGBHS y su propuesta memética que incluye ILS. 63
Figura 24: Gráfica NRMS vs Tiempo en la red McTrans del algoritmo SPEA-2 y su propuesta memética que incluye HC. 64
Figura 25: Gráfica NRMS vs Tiempo en la red McTrans del algoritmo SPEA-2 y su propuesta memética que incluye SA. 65
Figura 26: Gráfica NRMS vs Tiempo en la red McTrans del algoritmo SPEA-2 y su propuesta memética que incluye ILS. 65
Figura 27: Estadística GEH en la red McTrans de los algoritmos NSGA-II-ILS, MOGBHS y SPEA-2-ILS 67
Figura 28: Gráfica NRMS vs Tiempo en la red McTrans de la propuesta ganadora con los algoritmos del estado del arte. 68
Figura 29: Resultados de la prueba de Wilcoxon para la red McTrans. 69

Figura 30: Gráfica NRMS vs Tiempo en la red Reno del algoritmo NSGA-II y su propuesta memética que incluye HC. 71
Figura 31: Gráfica NRMS vs Tiempo en la red Reno del algoritmo NSGA-II y su propuesta memética que incluye SA. 72
Figura 32: Gráfica NRMS vs Tiempo en la red Reno del algoritmo NSGA-II y su propuesta memética que incluye ILS. 72
Figura 33: Gráfica NRMS vs Tiempo en la red Reno del algoritmo MOGBHS y su propuesta memética que incluye HC. 73
Figura 34: Gráfica NRMS vs Tiempo en la red Reno del algoritmo MOGBHS y su propuesta memética que incluye SA. 74
Figura 35: Gráfica NRMS vs Tiempo en la red Reno del algoritmo MOGBHS y su propuesta memética que incluye ILS. 75
Figura 36: Gráfica NRMS vs Tiempo en la red Reno del algoritmo SPEA-2 y su propuesta memética que incluye HC. 76
Figura 37: Gráfica NRMS vs Tiempo en la red Reno del algoritmo SPEA-2 y su propuesta memética que incluye SA. 76
Figura 38: Gráfica NRMS vs Tiempo en la red Reno del algoritmo SPEA-2 y su propuesta memética que incluye ILS. 77
Figura 39: Estadística GEH en la red Reno de los algoritmos NSGA-II-ILS, MOGBHS y SPEA-2-ILS. 78
Figura 40: Gráfica NRMS vs Tiempo en la red Reno de la propuesta ganadora con los algoritmos del estado del arte. 79
Figura 41: Resultados de la prueba de Wilcoxon para la red Reno. 80
Figura 42: Gráfica NRMS vs Tiempo en la red I-75 del algoritmo NSGA-II y su propuesta memética que incluye HC. 82
Figura 43: Gráfica NRMS vs Tiempo en la red I-75 del algoritmo NSGA-II y su propuesta memética que incluye SA. 82
Figura 44: Gráfica NRMS vs Tiempo en la red I-75 del algoritmo NSGA-II y su propuesta memética que incluye ILS. 83
Figura 45: Gráfica NRMS vs Tiempo en la red I-75 del algoritmo MOGBHS y su propuesta memética que incluye HC. 84
Figura 46: Gráfica NRMS vs Tiempo en la red I-75 del algoritmo MOGBHS y su propuesta memética que incluye SA. 84
Figura 47: Gráfica NRMS vs Tiempo en la red I-75 del algoritmo MOGBHS y su propuesta memética que incluye ILS. 85
Figura 48: Gráfica NRMS vs Tiempo en la red I-75 del algoritmo SPEA-2 y su propuesta memética que incluye HC. 86
Figura 49: Gráfica NRMS vs Tiempo en la red I-75 del algoritmo SPEA-2 y su propuesta memética que incluye SA. 86
Figura 50: Gráfica NRMS vs Tiempo en la red I-75 del algoritmo SPEA-2 y su propuesta memética que incluye ILS. 87
Figura 51: Estadística GEH en la red I-75 de los algoritmos NSGA-II-HC, MOGBHS y SPEA-2-HC. 88
Figura 52: Gráfica NRMS vs Tiempo en la red I75 de la propuesta ganadora con los algoritmos del estado del arte. 89
Figura 53: Resultados de la prueba de Wilcoxon para la red I75. 90

Figura 54: Frentes generados por los algoritmos NSGA-II-HC, MOGBHS y SPEA-2-HC en la red I-75. 92
Figura 55: Árbol de decisión valor promedio de NRMS. 93
Figura 56: Árbol de decisión tiempo promedio. 94

ÍNDICE DE TABLAS

Tabla 1: Vocabulario para MOGBHS. 55
Tabla 2: Vocabulario para NSGA-II con ILS. 55
Tabla 3: CA para MOGBHS. 56
Tabla 4: CA para NSGA-II con ILS. 57
Tabla 5: Resultados obtenidos por todos los algoritmos en la red McTrans. 66
Tabla 6: Resultados obtenidos por GASA y SPSA en la red McTrans. 69
Tabla 7: Resultados de la prueba de Friedman para la red McTrans. 70
Tabla 8: Resultados obtenidos por todos los algoritmos en la red Reno. 78
Tabla 9: Resultados obtenidos por GASA y SPSA en la red Reno. 79
Tabla 10: Resultados de la prueba de Friedman para la red Reno. 81
Tabla 11: Resultados obtenidos por todos los algoritmos en la red I-75. 88
Tabla 12: Resultados obtenidos por GASA y SPSA en la red I75. 89
Tabla 13: Resultados de la prueba de Friedman para la red I75. 91
Tabla 14: Categorización de los valores promedio de NRMS y Tiempo. 92

LISTADO DE ANEXOS

Anexo A: Prototipo Software, código fuente y ejecutable de la nueva versión de la herramienta Calibration Tools.
Anexo B: Manual de referencia CORSIM.
Anexo C: Artículo “Multi-objective Memetic Algorithm Based on NSGA-II and Simulated Annealing for Calibrating CORSIM Micro-Simulation Models of Vehicular Traffic Flow”.
Anexo D: Artículo “Algoritmo multi-objetivo basado en la mejor búsqueda global armónica para la calibración de modelos de micro-simulación de flujo de tráfico vehicular”.

CAPÍTULO 1

1. INTRODUCCIÓN

1.1 PLANTEAMIENTO DEL PROBLEMA

Los modelos de micro-simulación de tráfico vehicular permiten realizar un análisis minucioso del comportamiento de un sistema de transporte, para ello se deben tener en cuenta muchos aspectos que intervienen en un sistema vial, como por ejemplo, parámetros del comportamiento del conductor y del rendimiento del vehículo [1].

Existen muchos modelos de micro-simulación, como VISSIM, CORSIM, PARAMICS, entre otros. En estos modelos, hay una serie de parámetros que describen las características del flujo de tráfico, el comportamiento al volante y las operaciones del sistema de tráfico que tienen efectos significativos sobre los resultados de la simulación. En la presente investigación se utilizó el modelo de micro-simulación estocástico CORSIM, el cual consiste en un conjunto integrado de dos modelos de simulación microscópicas que representan todo el entorno de tráfico. NetSim que representa el tráfico en las vías urbanas y FRESIM que representa el tráfico en las autopistas [1]. CORSIM cuenta con diferentes variables de entrada, conocidas también como parámetros de calibración, los cuales permiten ser afinados con el fin de hacer coincidir el modelo con las condiciones de tráfico del mundo real. En CORSIM, particularmente, se cuenta con grupos de parámetros llamados “Record Type”, los cuales representan numerosas formas en las que una conducta, bien sea de los conductores o vehículos, se puede presentar sobre un link (unión entre dos nodos los cuales representan intersecciones de dos o más calles) durante una simulación.

Los datos de salida producidos por el modelo de micro-simulación CORSIM contienen numerosas medidas, dentro de estas están las medidas de volumen del flujo vehicular y la velocidad del flujo vehicular, que pueden ser usadas para realizar investigaciones que requieren que los datos de salida del modelo se asemejen lo mejor posible a los datos de campo o reales.

La identificación de parámetros sigue siendo una tarea difícil en el proceso de simulación [2], algunos motivos son: los parámetros no son directamente observables a partir de los datos comunes de tráfico, los datos no son transferibles a otras situaciones (diferentes lugares, momentos del día), entre otros. Por esta razón, es necesario realizar un proceso llamado calibración, que permite que dichos parámetros hagan que el modelo se asemeje más a la realidad.

Los modelos de micro-simulación CORSIM proporcionan un conjunto de valores predeterminados para cada parámetro y los usuarios pueden hacer uso de la simulación sin realizar un proceso de calibración a dichos parámetros, pero estos valores la mayoría de las veces no son representativos de la situación específica de tráfico que se está estudiando [3]. El objetivo de la calibración del modelo es definir

los valores para los parámetros de manera que el modelo simule el objeto de la vida real que se está estudiando tan estrechamente como sea posible [4].

Para la calibración de un modelo de simulación de tráfico, la dificultad reside en seleccionar la mejor combinación de los parámetros que se están calibrando. Sin embargo, todos los parámetros o un subconjunto amplio de estos deben ser calibrados de forma simultánea, y cada uno puede tener un rango de valores diferentes, que hacen que el proceso de calibración sea muy complicado y consuma mucho tiempo. Por lo tanto, para identificar el mejor conjunto de parámetros para el modelo, un algoritmo de optimización se presenta como una buena alternativa [3].

Recientemente, se han empezado a usar los algoritmos meméticos para resolver diversos problemas de optimización, mostrando que la estrategia de combinar la exploración, con la explotación y el conocimiento específico del problema pueden obtener mejores soluciones y en menor tiempo de ejecución [5]. Por esto, en [6] se reporta su uso para calibrar modelos de flujo vehicular, optimizando una sola función objetivo con resultados prometedores en comparación con el algoritmo Simultaneous Perturbation Stochastic Approximation (SPSA) [7], pero todavía esta propuesta no es apropiada para modelos de muy alta dimensionalidad y complejidad.

La mayoría de los problemas de optimización en el mundo real manejan múltiples objetivos, lo que significa que solucionarlos requiere optimizar dos o más funciones objetivos. Estos problemas se conocen como problemas de optimización multi-objetivo, donde el óptimo no es una solución única (problemas mono-objetivo), sino un conjunto de soluciones [8].

Cuando los tomadores de decisiones pueden explorar y visualizar diferentes soluciones óptimas de varios objetivos en conflicto les ayuda a obtener una mejor comprensión del problema y de las potenciales soluciones que están disponibles [9]. Las soluciones de un problema multi-objetivo son múltiples y son denominadas óptimos de Pareto. Una solución es un óptimo de Pareto si no existe otra solución que mejore alguno de los objetivos sin empeorar alguno de los otros. Un problema multi-objetivo puede tener muchos, e incluso infinitos óptimos de Pareto los cuales conforman el llamado frente de Pareto. Generalmente, decidir cuál de estos óptimos es la mejor solución al problema es una tarea única de los tomadores de decisiones, quienes deben incluir uno o más criterios de desempate [10].

Los algoritmos multi-objetivo siguen creciendo en popularidad en muchos campos de la ingeniería [9] [11], debido a su capacidad para manejar los problemas con diferentes tipos de variables de decisión. Uno de los algoritmos de optimización multi-objetivo más utilizados y competitivos [8] [12] [13] es el algoritmo genético basado en ordenamiento no dominado-II (NSGA-II) [14], el cual es conocido en la comunidad científica por su buen desempeño y por los pocos cambios necesarios respecto al algoritmo mono-objetivo del que se inspiró para ser implementado apropiadamente [15]. Sin embargo, a partir de los estudios que comparan diferentes algoritmos de calibración automática multi-objetivo [16], se observa que no hay un

algoritmo que sea superior en todos los casos. En consecuencia, en el presente trabajo de investigación se propone, además de la adaptación de NSGA-II al problema de calibración, la adaptación del algoritmo multi-objetivo basado en la mejor búsqueda global armónica (MOGBHS) al mismo problema. MOGBHS es una modificación del algoritmo mono-objetivo de la mejor búsqueda armónica global (GBHS por sus siglas en inglés) el cual se ha destacado por sus buenos resultados [17] [18] [19]. El algoritmo MOGHBS fue propuesto en [20] para la definición de rutas y horarios en un sistema integrado masivo de pasajeros y fue comparado con el algoritmo NSGA-II, mostrando mejores resultados en ese problema específico. En la adaptación de los algoritmos se agregó conocimiento del problema buscando que los algoritmos multi-objetivos encuentren mejores soluciones en menos tiempo [21].

Según [22], el volumen y la velocidad son los parámetros más relevantes en un modelo de micro-simulación de flujo de tráfico vehicular. Al igual que en [6, 22] en el presente trabajo de investigación se determinó que el modelo de simulación de flujo de tráfico vehicular CORSIM se encontraba calibrado correctamente si éste cumplía con las reglas y procedimientos descritos en el manual de CORSIM [1]; dado que en [4] se afirma que es recomendable tener diversas funciones objetivo cuando se trabaja con modelos que simulan el mundo real, en este trabajo se establece el problema de calibración en un marco multi-objetivo, optimizando de manera simultánea los parámetros de velocidad y volumen.

Teniendo en cuenta lo anterior, en este trabajo se planteó la siguiente pregunta de investigación: **¿Cómo realizar el proceso de calibración de un modelo de flujo de tráfico vehicular CORSIM buscando obtener mejores resultados en calidad y en menor tiempo?**

El enfoque de solución a esta pregunta de investigación se centró en la adaptación de los algoritmos meméticos multi-objetivos NSGA-II y MOGBHS al problema específico de calibración.

En el presente proyecto de investigación se compararon los algoritmos propuestos con el algoritmo SPSA, dado que dicho algoritmo ha demostrado algunas mejoras en cuanto a tiempo y recursos computacionales en el campo de la calibración automatizada de modelos [23], la comparación también se hizo con el algoritmo multi-objetivo SPEA-2 [24] el cuál se presenta como una de las técnicas multi-objetivo más utilizadas en el estado del arte y el algoritmo GASA un algoritmo genético mono-objetivo que usa recocido simulado, propuesto en [6] y usado para la calibración de modelos de micro-simulación de tráfico vehicular CORSIM. Esta comparativa se realizó utilizando la métrica Normalized Root Mean Square (NRMS) [25] reconocida en el estado del arte.

1.2 JUSTIFICACIÓN

Desde el punto de vista de investigación, las contribuciones del presente proyecto se enfocaron en la generación de nuevo conocimiento relacionado con la calibración de modelos de micro-simulación de flujo de tráfico vehicular CORSIM utilizando

algoritmos multi-objetivo y definiendo si el nuevo algoritmo MOGBHS lograba ser más eficiente (tiempo de ejecución) y obtener mejores resultados que el algoritmo NSGA-II. Además, también propone la obtención de nuevo conocimiento relacionado con los resultados de la comparación con el algoritmo SPSA y el algoritmo memético mono-objetivo propuesto en [6], denominado GASA, los cuales han demostrado tener buenos resultados en el proceso de calibración. De la misma manera se realizó una comparación con el algoritmo multi-objetivo SPEA-2 el cual es muy utilizado en el estado del arte en diversos problemas que requieren de la optimización de dos o más objetivos, esto último va más allá de los objetivos inicialmente definidos en el proyecto.

Desde el punto de vista de innovación, en este proyecto se adicionó a una herramienta software existente, denominada Calibration Tools, tres algoritmos de optimización multi-objetivo (NSGA-II, MOGBHS y SPEA-2) hibridados con tres algoritmos de búsqueda local (Subiendo la colina, Recocido simulado y Búsqueda local iterada), que permiten usar diversos algoritmos meméticos multi-objetivo para la calibración de modelos de tráfico vehicular CORSIM de carreteras de Estados Unidos. Además, estos algoritmos pueden ser usados en otros problemas reales de tráfico vehicular en el mundo.

1.3 OBJETIVOS

A continuación, se presentan los objetivos como fueron aprobados en el anteproyecto por parte del Consejo de Facultad, de la Facultad de Ingeniería Electrónica y Telecomunicaciones.

1.3.1 OBJETIVO GENERAL

Proponer un algoritmo memético multi-objetivo para calibrar modelos de micro-simulación de flujo de tráfico vehicular CORSIM.

1.3.2 OBJETIVOS ESPECÍFICOS

- Adaptar los algoritmos multi-objetivo NSGA-II y MOGBHS para que permitan calibrar modelos de micro-simulación de flujo de tráfico vehicular CORSIM, usando los componentes de la función objetivo propuesta en [6].
- Adaptar un algoritmo de optimización local (Ascenso a la colina, Recocido simulado y Búsqueda local iterada) para que incluya conocimiento del problema en los algoritmos multi-objetivo adaptados.
- Evaluar la calibración de modelos de micro-simulación de flujo de tráfico vehicular realizada por los algoritmos meméticos multi-objetivo propuestos, utilizando las directrices de la FHWA [1], y finalmente comparar los resultados obtenidos con el algoritmo SPSA [23] y el memético mono-objetivo propuesto en [6] en tiempo y mejores soluciones encontradas.

1.4 RESULTADOS OBTENIDOS

- Monografía con el resumen del proyecto: corresponde al presente documento que contiene el estado del arte referente a diversas aplicaciones de los algoritmos multi-objetivo así como el proceso de calibración de modelos de tráfico vehicular, se presenta en detalle los dos algoritmos multi-objetivo (NSGA-II, MOGBHS) así como los tres algoritmos de optimización local y su hibridación para la obtención del conjunto de algoritmos meméticos multi-objetivo que soportan el proceso de calibración, finalmente muestra los resultados de la evaluación de los algoritmos y su comparación con los algoritmos SPSA, SPEA-2 y GASA [6].
- Prototipo Software, código fuente, documentación y ejecutable de la nueva versión de la herramienta Calibration Tools con la inclusión de los tres algoritmos meméticos multi-objetivo desarrollados en la presente investigación.
- Un artículo publicado en evento internacional y revista internacional indexada categoría A1 según PUBLINDEX de Colciencias y Q3 en SJR con la siguiente referencia: Cobos, C., Erazo, C., Luna, J., Mendoza, M., Gaviria, C., Arteaga, C., Paz, A. Ordoñez. Multi-Objective Memetic Algorithm based on NSGA-II and Simulated Annealing for Calibrating CORSIM Micro-Simulation Models of Vehicular Traffic Flow. MAEB 2016 - XI Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados. Advances in Artificial Intelligence: Proceedings of the 17th Conference of the Spanish Association for Artificial Intelligence, CAEPIA 2016. O. Luaces et al. (Eds.). Springer International Publishing Switzerland. pp. 468-476. Series: Lecture Notes in Computer Science (LNCS), Subseries: Lecture Notes in Artificial Intelligence (LNAI). Salamanca, Spain, September 14-16, 2016, vol. 9868. ISSN: 0302-9743 (Print) 1611-3349 (Online). Available online at http://dx.doi.org/10.1007/978-3-319-44636-3_44
- Un nuevo artículo que resume el desarrollo de toda la investigación y los resultados obtenidos, el cuál será enviado a revista internacional indexada JCR (ISI) a finales del 2016. La versión en español se presenta como anexo de este documento.

1.5 ESTRUCTURA DE LA MONOGRAFÍA

A continuación, se describe de manera general el contenido y organización de la presente monografía.

CAPÍTULO 1: Introducción

En este capítulo se presenta una introducción al tema de investigación, la pregunta de investigación, la justificación del problema de investigación, los objetivos (general y específicos) aprobados y finalmente la organización del documento.

CAPÍTULO 2: Contexto teórico y Estado del arte

En la primera parte de este capítulo se presenta una descripción y explicación detallada de los conceptos (contexto teórico) utilizados en el presente proyecto de

investigación. En la segunda parte se presentan las investigaciones más notables acerca del problema de investigación planteado.

CAPÍTULO 3: Propuesta

En este capítulo se hace una descripción detallada de la solución al problema de calibración de modelos de micro-simulación de tráfico vehicular CORSIM, también se presentan los algoritmos de búsqueda global (NSGA-II, MOGBHS y SPEA-2) y búsqueda local (subiendo la colina, recocido simulado y búsqueda local iterada) propuestos para resolver el problema en un marco memético multi-objetivo.

CAPÍTULO 4: Experimentación

En este capítulo se describen los modelos utilizados para la evaluación de los algoritmos propuestos, se presentan los algoritmos del estado del arte con los que se compararon los algoritmos propuestos, se muestran las medidas con las que se realizó su comparación, se hace una descripción del proceso de afinamiento de los parámetros, se presentan los resultados obtenidos por los algoritmos propuestos y los del estado del arte y finalmente, se presenta el análisis de sensibilidad del algoritmo ganador.

CAPÍTULO 5: Conclusiones y Trabajos a futuro

En la primera parte de este capítulo se sintetizan las conclusiones a las que se llegó después de terminar el proyecto de investigación. En la segunda parte se presentan posibles mejoras o adiciones que se puedan realizar al enfoque desarrollado en el presente trabajo.

CAPÍTULO 6: Bibliografía

Este capítulo contiene los artículos y libros de referencia u otros documentos consultados en la realización del proyecto.

CAPÍTULO 2

2. CONTEXTO TEÓRICO

2.1 INTRODUCCIÓN

En esta sección se presentan algunos conceptos utilizados en este proyecto con el propósito de facilitar un mayor entendimiento del mismo.

2.2 SIMULACIÓN Y MICRO-SIMULACIÓN

La simulación se puede definir como el proceso de diseñar y desarrollar un modelo computarizado de un sistema o proceso y conducir experimentos con este modelo con el propósito de entender el comportamiento del sistema o evaluar varias estrategias con las cuales se pueda operar el mismo. La simulación es una herramienta poderosa para los tomadores de decisiones ya que hace posible el estudio, análisis y evaluación de situaciones que sin ella no sería posible realizar [26]. Hay varios tipos de simulación, entre ellos el enfoque estocástico, que se basa en la simulación de la aleatoriedad de sucesos y/o entidades que se presentan en un entorno real, una característica clave de este enfoque es que la fuente de aleatoriedad está bajo el control del experimentador [27].

Dentro del campo de la simulación se contempla la micro-simulación o también llamada simulación microscópica que se puede describir como un conjunto de herramientas que facilitan un estudio detallado de un modelo. La esencia de éste enfoque es el uso de la aleatorización en la asignación de valores a las unidades estudiadas en el modelo, es decir, en la "predicción" y el uso de unidades individuales de análisis. La especificación de micro-simulación contempla un modelo y los valores de los parámetros que son obtenidos por estimación estadística u otros medios [28].

En general, la micro-simulación consiste en extraer una muestra de las realizaciones de un proceso estocástico especificado previamente, por lo que el proceso de micro-simulación presupone la existencia de un modelo, así como la disponibilidad de valores específicos para todos sus parámetros. Una vez más, el modelo (el proceso estocástico pre-especificado) debe ser conocido de antemano. Los datos generados se verán como los datos "reales", y pueden, por lo tanto, ser analizados y resumidos de igual manera que los verdaderos datos reales [28].

2.3 MODELO DE MICRO-SIMULACIÓN CORSIM

CORSIM (CORridor SIMulation) es un software de simulación microscópica desarrollado por la Federal Highway Administration (FHWA) a mediados de la década de 1970 actualmente es mantenido y distribuido por McTrans en la Universidad de Florida. Se ejecuta dentro de un entorno de software denominado Sistema Integrado de Software de Tráfico (TSIS por sus siglas en inglés) que proporciona una interfaz basada en Windows y un entorno para la ejecución del

modelo. Un elemento clave de TSIS es el procesador de salida TRAFVU, que permite al analista ver gráficamente la red y evaluar su eficiencia mediante la animación.

CORSIM fue diseñado para estudiar las operaciones de tráfico de las autopistas y las calles de la superficie, éste incluye dos modelos predecesores: FRESIM (FREway SIMulation) que es un modelo microscópico que simula el tráfico en las autopistas, y NETSIM (NETwork SIMulation) que se encarga de simular el tráfico urbano. Las capacidades de CORSIM incluyen simulación de diferentes controles de intersección y una amplia gama de condiciones de flujo de tráfico [29]. CORSIM es un modelo estocástico que incorpora procesos aleatorios para modelar: conductores, vehículos y los comportamientos del sistema de tráfico e interacciones complejas. Las redes CORSIM se basan en una representación enlace-nodo. Cada enlace representa un segmento de carretera en una sola dirección, mientras que los nodos representan las intersecciones y puntos de entrada y salida [30].

Los archivos que recibe como entrada CORSIM contienen la información utilizada para definir un modelo y poder conducir la simulación del mismo (extensión ".trf"), a su vez, estos archivos están compuestos de unos conjuntos de parámetros llamados "Record Type" los cuales brindan la información acerca de los diversos comportamientos presentes en la simulación del modelo.

Los archivos de salida generados por CORSIM contienen información resumida de las estadísticas obtenidas de la simulación sobre diferentes nodos y enlaces (extensión ".out"), con estos archivos se puede observar, por ejemplo, el tiempo que perteneció un vehículo en un determinado enlace.

2.4 CALIBRACIÓN DE MODELOS DE SIMULACIÓN DE TRÁFICO VEHICULAR

La calibración se puede definir como el proceso iterativo de alterar los parámetros de un modelo dentro de un rango físico posible, buscando aquellos valores que minimizan la diferencia entre lo simulado y un conjunto de datos reales. El proceso de calibración se puede realizar de dos diferentes maneras. La primera de ellas es llamada calibración manual, esta técnica se basa en los resultados de análisis de sensibilidad, y utiliza un método de ensayo y error para obtener un conjunto de parámetros para cada evento por separado [31]. La segunda forma es el procedimiento de calibración automática que se basa en simulaciones de los modelos, este procedimiento tiene como objetivo explorar todo el espacio de parámetros, especialmente alrededor de los límites obtenidos en la calibración manual [4]. La calibración automática ajusta los parámetros automáticamente de acuerdo con un esquema de búsqueda especificada para optimizar medidas numéricas o índices que reflejan el grado de ajuste de los resultados de la simulación del modelo a los datos reales. La calibración automática ha ganado cada vez más popularidad en las últimas décadas, ya que resuelve los principales inconvenientes de los métodos manuales que son subjetivos, tediosos, dependen mucho de la experiencia de los modeladores y que necesitan gran cantidad de mano de obra [32] [33] [34].

La técnica de calibración dentro de modelos de tráfico vehicular consiste en comparar los valores obtenidos por un instrumento de medición, por ejemplo, sensores ubicados sobre una vía que realizan medidas de diferentes métricas (como velocidad de los vehículos); dicha vía estaría representada en el modelo de simulación que arroja los valores con los que se va a comparar. Posteriormente mediante diferentes técnicas se modifican los valores que recibe como entrada el modelo, con el fin de que la salida de la simulación se asemeje cada vez más a los valores que se obtienen, en este caso los capturados por los sensores. El proceso de calibración se hace necesario debido a que el funcionamiento de un sistema de tráfico está bajo la influencia de diversos aspectos de la conducta humana [35] de los cuales algunos de ellos son imposibles de medir.

En un modelo de simulación de tráfico vehicular dos grupos de parámetros requieren ser calibrados. Los parámetros del comportamiento del conductor, los cuales incluyen la aceleración, cambio de carril, y los modelos de intersecciones. Y los parámetros del comportamiento del viaje los cuales están representados por la ruta a modelar. La calibración de modelos de tráfico vehicular no es una tarea trivial, debido a que los datos normalmente disponibles son las medidas globales de las características de tráfico (por ejemplo, los flujos de velocidad, los tiempos de viaje, y la longitud de colas), que son los resultados emergentes de las interacciones entre los diversos comportamientos de los vehículos individuales. Por lo tanto, este tipo de datos no soporta la calibración independiente de los diversos modelos que componen el simulador de tráfico [36].

La calibración de modelos de simulación de tráfico vehicular puede ser vista como un problema de optimización con un gran espacio de búsqueda y a menudo como un problema no lineal y altamente complejo [37].

2.5 ALGORITMO DE OPTIMIZACIÓN

Cuando en un problema a resolver se cuenta con más de dos parámetros, sus interacciones pueden ser muy difíciles de predecir y la tarea de diseño se hace difícil. Tales problemas a menudo se pueden tratar en términos de un problema de optimización basados en alguna medida de mérito o función objetivo que se maximiza o minimiza mediante la alteración de los parámetros de diseño, mientras se debe cumplir con diversas restricciones [38]. Los métodos para resolver el problema general de optimización se han estudiado durante muchos años y existe una literatura considerable sobre ello (como se muestra en la sección posterior).

En cualquier espacio de búsqueda de un problema de optimización dado, hay dos tipos de óptimos: globales y locales. La definición de óptimo también cambia dependiendo de si el objetivo es maximizar o minimizar. En el caso de minimización, un mínimo global indica la ubicación de un punto en el espacio de búsqueda con la evaluación más baja de la función objetivo. Un mínimo local, por otro lado, representa el valor de la función objetivo más bajo sobre una porción limitada del espacio de búsqueda [39].

El problema de optimización se puede resumir como el proceso de encontrar el conjunto de parámetros que minimiza o maximiza, tal sea el caso, una función objetivo $f(X_0,\ldots, X_n)$ que también se conoce como la función de aptitud en los algoritmos evolutivos. Un proceso de optimización global es aquel que requiere la llegada a la mejor decisión en cualquier conjunto dado de circunstancias [40].

2.6 ALGORITMO MEMÉTICO

El concepto de un algoritmo memético se introdujo por primera vez por Moscato y Norman [41] para describir los algoritmos evolutivos en los que la búsqueda local se utiliza en gran medida, este término está motivado por la noción de Richard Dawkins de un meme como una unidad de información que se reproduce a sí mismo como por ejemplo, el intercambio de ideas entre las personas. Un meme difiere de un gen en que antes de que un meme sea transmitido, suele ser adaptado por el individuo que lo transmite según como el individuo entiende y procesa el meme, mientras que los genes se pasan inalterados [42].

Los algoritmos meméticos son meta-heurísticas basadas en población, el funcionamiento de estos algoritmos se realiza mediante la combinación de sus dos principales componentes, el primero de ellos es la exploración donde un algoritmo realiza una amplia búsqueda sobre todo el espacio de soluciones para determinar las regiones interesantes que pueden contener soluciones óptimas, con ello se logra que el espacio de posibles soluciones se reduzca al subespacio de óptimos locales. Posterior a esto, un algoritmo de explotación o también conocido como algoritmo de búsqueda local realiza una búsqueda exhaustiva sobre la región escogida previamente por el algoritmo de exploración con el objetivo de encontrar el mejor resultado perteneciente a dicha región. La inclusión de la búsqueda local tiene un costo computacional inevitable, pero esto puede ser justificado por la reducción en el tiempo de búsqueda con el fin de encontrar la solución óptima [42]. Además, los algoritmos meméticos incorporan conocimiento del problema a través de los memes, esto puede ser útil para hallar eficientemente mejores soluciones a un problema de optimización. Los algoritmos meméticos intentan explotar todo el conocimiento que se tenga sobre el problema de optimización. Esta explotación del conocimiento puede llevarse a cabo de diferentes formas, por ejemplo, introducir conocimiento del problema mediante el empleo de heurísticas constructivas en los operadores de inicialización usados para la generación de la población inicial o en los operadores de reproducción, entre otros. [43].

Los algoritmos meméticos logran mantener un equilibrio entre la exploración y explotación, que es a menudo crucial para el éxito de los procesos de búsqueda y optimización. Estudios recientes sobre los algoritmos meméticos han revelado sus éxitos en una amplia variedad de problemas del mundo real, mostrando que no sólo convergen hacía soluciones de alta calidad, sino también realizan la búsqueda de manera más eficiente que otros algoritmos convencionales [44].

2.7 ALGORITMO MULTI-OBJETIVO

La mayoría de los problemas de optimización en el mundo real son multi-objetivo, lo que significa que su solución requiere optimizar dos o más funciones u objetivos que en muchos casos son contradictorios. Estos problemas se conocen como problemas de optimización multi-objetivo. El óptimo buscado en este tipo de problemas no es una solución única como en el caso de los problemas mono-objetivo, sino un conjunto de soluciones óptimas, conocidas como el conjunto óptimo de Pareto [8].

Cualquier elemento perteneciente a este conjunto óptimo de Pareto no es mejor que los demás en todos los objetivos, por lo tanto, ninguno de ellos puede ser considerado como el mejor sin una información adicional. Este conjunto de soluciones es entregado al tomador de decisiones, que tiene que elegir la mejor solución de acuerdo a sus preferencias. El concepto de Pareto es introducido en 1906 en el contexto de la economía haciendo referencia a aquella situación en la cual se cumple que no es posible beneficiar a una persona sin perjudicar a otra. Posteriormente Zadeh (1963) y Stadler (1979) comenzaron a aplicar el concepto de óptimo de Pareto a los campos de la ingeniería. Una definición más formal indica que una solución es óptima de Pareto si ninguna otra solución factible existe que mejore alguno de los objetivos sin causar un empeoramiento simultáneo en al menos otro objetivo. Cuando una solución es un óptimo de Pareto, implica que es una solución no dominada por ninguna otra, se dice que una solución domina a otra si ésta presenta una mejora en al menos uno de los objetivos presentes en el problema y es, por lo menos, igual en los otros [45].

La obtención del frente de Pareto es el principal objetivo de la optimización multi-objetivo. En teoría, un frente de Pareto podría contener un gran número de soluciones (incluso un número infinito). Sin embargo, en la práctica, una solución aproximada utilizable sólo contendrá un número limitado de ellos, por lo tanto, un objetivo importante es que todas las soluciones deben estar lo más cerca posible del frente de Pareto (convergencia) y dicho frente debe estar repartido de manera uniforme (diversidad), de otro modo, no serían muy útiles para el tomador de decisiones. El hecho de que una solución presente cercanía con el frente de Pareto asegura de que se está tratando con una solución óptima, mientras que una dispersión uniforme de las soluciones significa que se ha logrado una buena exploración del espacio de búsqueda y no hay regiones que están sin explorar [8].

A modo de ejemplo, la Figura 1 muestra el frente de Pareto dividido en dos conjuntos. Los elementos más alejados entre sí (puntos rojos), tendrán prioridad sobre los demás elementos del frente (puntos negros), puesto que estos aportan mayor diversidad a la población, es decir están más dispersos en el espacio de búsqueda.

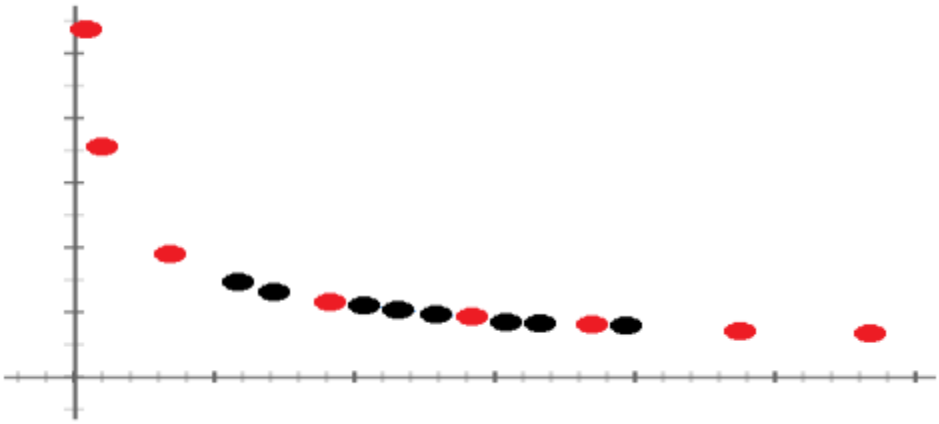

Figura 1: Ejemplo del frente de Pareto.

2.7.1 ALGORITMO NSGA-II

Los algoritmos multi-objetivos requieren para su desarrollo de métodos matemáticos de optimización sobre una población de soluciones, por lo que se ha encontrado en los algoritmos genéticos una propuesta prometedora, dadas sus características de diversidad y confiabilidad. Sin embargo se debe mantener una mente abierta y posibilitar el uso de otros mecanismos [15]. El algoritmo genético basado en ordenamiento no dominado (NSGA) fue de los primeros algoritmos multi-objetivo en ser desarrollado a partir de un algoritmo genético, pero este presenta las siguientes falencias:

- Alta complejidad computacional en el procedimiento del ordenamiento de no-dominancia: Su complejidad computacional es de $O\,(MN^3)$ para cada generación (donde M es el número de objetivos y N es el tamaño de la población) debido al procedimiento de selección de las soluciones no dominadas en cada generación, haciendo que NSGA sea computacionalmente costoso para grandes tamaños de población.
- Falta de elitismo: Diversas investigaciones demuestran que el elitismo puede acelerar el rendimiento de los algoritmos genéticos significativamente, lo que también puede ayudar a prevenir la pérdida de buenas soluciones, una vez que se encuentran.
- Necesidad de especificar un parámetro para garantizar la diversidad en la población: Esto con el fin de obtener una amplia variedad en las soluciones.

NSGA-II [14] es propuesto por Deb en 2002 con el propósito de mejorar las falencias que presentaba NSGA. La primera de ellas utilizando un enfoque rápido de clasificación de no dominancia, reduciendo así la complejidad computacional en el procedimiento de ordenamiento a $O\,(MN^2)$, acelerando la búsqueda y evitando la pérdida de soluciones prometedoras ya obtenidas. También añade elitismo seleccionando los mejores miembros de la población convirtiéndose en los padres de la siguiente generación. Por último, define un esquema de diversidad llamado distancia de Crowding (en español distancia de apiñamiento) que sustituye el "sharing" utilizado por NSGA cuyo objetivo es introducir diversidad en las soluciones

del problema. Cuando existan dos o más soluciones con iguales valores de aptitud o no-dominancia se escoge una solución que esté menos apiñada en el espacio de soluciones, esto es, seleccionar una solución que sea más diferente a las demás y que permita explorar mejor el espacio de soluciones [46].

2.7.2 ALGORITMO MOGBHS

El algoritmo de búsqueda armónica (HS por sus siglas en inglés) se ha desarrollado recientemente [47] como una analogía con el proceso de improvisación musical, donde los músicos de Jazz tratan de pulir sus tonos para obtener una mejor armonía. El proceso de improvisación musical es similar al proceso de optimización pues ambos procesos buscan encontrar una solución óptima. El tono de cada instrumento musical determina la calidad de la armonía, al igual que la función objetivo asignado al conjunto de variables. El algoritmo HS se ha aplicado con éxito a muchos problemas de optimización del mundo real y ha demostrado ser mejor que otros algoritmos evolutivos en términos de calidad de la solución [17].

En comparación con otras metaheurísticas de la literatura, el algoritmo HS requiere menos requerimientos matemáticos y se puede adaptar fácilmente para resolver diversos tipos de problemas de optimización de ingeniería. El algoritmo HS es bueno en la identificación de las regiones de alto rendimiento en el espacio de soluciones en un tiempo razonable. Sin embargo, no es eficiente en la realización de la búsqueda local en aplicaciones de optimización numérica [48]. Por lo tanto, algunas modificaciones se han desarrollado para mejorar la precisión de la solución y la velocidad de convergencia.

Mahdavi et al. [48] presenta un algoritmo HS mejorado, denominado IHS, mediante la introducción de una estrategia para ajustar dinámicamente los parámetros claves de HS. Mientras que Omran y Mahdavi [49] propusieron un algoritmo denominado mejor búsqueda armónica global (GBHS por sus siglas en inglés), incluyendo en HS la técnica de enjambre de partículas y mostrando que el algoritmo GBHS obtiene mejores soluciones que HS e IHS [50].

El algoritmo multi-objetivo basado en la mejor búsqueda global armónica o Multi-Objetive Global Best Harmony Search (MOGBHS) propuesto por Ruano, Cobos y Torres-Jimenez en el marco del programa de Maestría en computación de la Universidad del Cauca, adaptado en este proyecto, cuenta con dos componentes principales: 1) GBHS como una estrategia de optimización heurística (búsqueda global y local en el espacio de soluciones), y 2) clasificación de soluciones no dominadas para clasificar las soluciones basadas en múltiples objetivos. El algoritmo GBHS es responsable de la generación y evolución de los individuos en cada generación. El segundo componente de MOGBHS lleva a cabo la clasificación de las soluciones utilizando el concepto de soluciones no dominadas para construir un frente de Pareto [51].

2.7.3 ALGORITMO SPEA-2

Después de los primeros estudios sobre la optimización multi-objetivo evolutiva (EMO), se propusieron una serie de técnicas basadas en el concepto de Pareto, como: MOGA en 1993, NPGA y NSGA en 1994. Posteriormente se presenta el algoritmo evolutivo de fuerza de Pareto (SPEA), su funcionamiento se basa en el uso de elitismo mediante un archivo, o población externa, en donde se van guardando las mejores soluciones, es decir, las pertenecientes al frente de Pareto (selección de ambiente) y el uso del concepto de selección de reproducción (Selección por torneo sobre los individuos de la población principal y la externa, con el objetivo de reproducirse). Este algoritmo obtuvo buenos resultados en comparación con otras técnicas multi-objetivo [52], sin embargo, presenta algunas deficiencias que hacen que su rendimiento no sea óptimo, como:

- Asignación de aptitud: Los individuos que están dominados por los mismos miembros del archivo tienen valores idénticos de aptitud. Eso significa que en el caso de que el archivo contenga sólo una sola solución, todos los miembros de la población tienen el mismo valor independientemente de que se dominen entre sí o no. Como consecuencia, en este caso particular, SPEA se comporta como un algoritmo de búsqueda aleatoria.
- Falta de un estimador de densidad: Si muchos individuos de la generación actual no se dominan el uno al otro, ninguna o muy poca información se puede obtener a partir de la relación de dominancia sobre el orden en el que deben ir estas soluciones. El método de agrupación hace uso de ésta información, pero sólo con respecto al archivo y no a la población.
- Truncamiento del archivo: Aunque la técnica de agrupación utilizada en SPEA es capaz de reducir el conjunto de no dominados sin destruir sus características, con esto se corre el riesgo de perder soluciones externas.

Para contrarrestar estas deficiencias, en [24] proponen una versión mejorada denominada SPEA-2, cuyas diferencias son las siguientes:

- Define un nuevo esquema de asignación de aptitud, en donde para cada individuo se tiene en cuenta el número de individuos que domina y que le dominan.
- Incorpora una técnica de estimación de la densidad del vecino más cercano, está técnica permite una orientación más precisa del proceso de búsqueda. (similar al operador de Crowding de NSGA-II).
- Implementa un método de truncamiento que sustituye la agrupación utilizada por SPEA, garantizando la preservación de soluciones del contorno.
- Sólo los miembros del archivo participan en el proceso de selección de reproducción.

2.8 ALGORITMOS DE BÚSQUEDA LOCAL

Un enfoque prometedor para mejorar la velocidad de convergencia para el frente de Pareto es el uso de algoritmos de búsqueda local [53]. La hibridación de los

algoritmos de búsqueda global con algoritmos de búsqueda local ya ha sido investigada en diversos problemas de optimización de un solo objetivo. Tal algoritmo híbrido se refiere a menudo como un algoritmo memético, concepto explicado en la sección 2.6. La función que desempeña el algoritmo de búsqueda local es la de mejorar algunos o todos los individuos que componen una determinada población. La búsqueda local y la selección de su regla de búsqueda en el vecindario son fundamentales para el éxito dentro de los esquemas de búsqueda global [54]. A continuación se presenta la descripción detallada de los tres algoritmos de búsqueda local utilizados en esta investigación, se optó por la utilización de estos algoritmos dado que: Ascenso a la colina cuenta con una estrategia de explotar sobre la mejor solución, esto lo convierte en un método de búsqueda local prometedor, de igual manera el algoritmo de búsqueda local iterada es escogido dado que este se presenta como una versión mejorada de la heurística ascenso a la colina, con una estrategia para salir de óptimos locales. Por último, el algoritmo recocido simulado es escogido dado que este ya ha sido utilizado como método de explotación para el problema de calibración de flujo de tráfico vehicular.

2.8.1 ALGORITMO DE ASCENSO A LA COLINA

Ascenso a la colina (HC por sus siglas en inglés) es un algoritmo de búsqueda local y se basa en mejoras incrementales de las soluciones de la siguiente manera: comienza con una solución que puede ser generada de forma aleatoria y es considerada como la solución actual en el espacio de búsqueda. El algoritmo examina los vecinos de la solución actual y si un vecino es mejor que la solución actual, entonces se convierte en la nueva solución actual; el algoritmo sigue moviéndose de una solución a otra en el espacio de búsqueda hasta que no encuentra ninguna otra mejora o se cumpla el criterio de parada previamente establecido [55]. Una de las desventajas de esta técnica es que por lo general se estanca en óptimos locales por lo que se han adaptado algunas estrategias para evitar dicho problema, algunas de ellas son:

- Volver a un estado anterior y seguir el proceso en otra dirección.
- Saltar a una nueva solución, posiblemente, "muy lejos" de la solución actual.
- Considerar varias direcciones de búsqueda en el espacio de solución al mismo tiempo, por ejemplo, dividir el espacio de búsqueda en regiones y explorar las más prometedoras.

2.8.2 ALGORITMO DE RECOCIDO SIMULADO

El algoritmo de recocido simulado (SA por sus siglas en inglés) es una técnica que hace una analogía física del proceso de calentamiento y después de enfriamiento lento de un metal para obtener un cristalizado fuerte [56]. SA es un algoritmo de búsqueda local que evita quedar atrapado en óptimos locales. La técnica primero genera una solución inicial normalmente obtenida al azar, luego se genera un vecino de esta solución y se calcula el cambio en el costo que depende de la diferencia entre los valores de función correspondientes y en un parámetro global T (llamado temperatura), que se reduce gradualmente durante el proceso iterativo de búsqueda

de la solución. Si se encuentra una reducción en el costo, la solución actual se sustituye por el vecino generado, de otro modo una estrategia basada en el aprendizaje Boltzviano es utilizada por el algoritmo para determinar si la nueva solución es aceptada como la solución actual del problema. El proceso se repite hasta que no se encuentren nuevas mejoras en la zona de la solución actual por lo que el algoritmo termina en un óptimo local.

Una desventaja de los algoritmos de búsqueda local es que el óptimo local puede ser encontrado lejos de cualquier óptimo global. Una forma de mejorar la solución es ejecutar el algoritmo varias veces a partir de diferentes soluciones iniciales, el algoritmo SA presenta una estrategia diferente a la anterior para evitar ese problema, dicha estrategia consiste en aceptar soluciones que empeoran el valor de la función objetivo. La aceptación o el rechazo de una mala solución es determinada por una secuencia de números aleatorios, pero con una probabilidad controlada. Esta es conocida como la función de aceptación y normalmente se establece en $e^{-\frac{\delta}{T}}$ donde δ es una función de costo y T es un parámetro de control que corresponde a la temperatura en la analogía con el recocido físico [57]. El algoritmo SA utilizado en esta investigación tiene sus bases en el propuesto en [58] para resolver problemas multi-objetivo; en el hacen uso del concepto de dominancia para determinar si una solución es tomada como óptima del problema.

2.8.3 ALGORITMO DE BÚSQUEDA LOCAL ITERADA

El algoritmo de búsqueda local iterada (ILS) por sus siglas en ingles es una técnica que ha existido desde 1980. Básicamente se trata de una versión más inteligente del algoritmo de ascenso a la colina con reinicios aleatorios. La idea principal es hacer una búsqueda más inteligente a través del espacio de óptimos locales. Se centra en realizar la búsqueda de un óptimo local al azar, y luego otro, y luego otro, y así sucesivamente, y, finalmente, volver al mejor óptimo que se haya descubierto (lo ideal, es que éste sea un óptimo global).

La idea central de ILS es que a menudo se puede encontrar mejores óptimos locales cerca a la solución actual, caminando de óptimo local en óptimo local en lugar de simplemente probar nuevas ubicaciones completamente al azar. El algoritmo ILS logra esto con dos procedimientos fundamentales. En primer lugar, el ILS no realiza reinicios al azar, más bien, se mantiene una “base” de óptimos locales, y selecciona nuevas ubicaciones de reinicio que se encuentran cerca, aunque no en exceso, de las proximidades de la “base”. Se desea reiniciar lo suficientemente lejos de la base actual para terminar en un nuevo óptimo local, pero no tan lejos como para estar recogiendo nuevas ubicaciones de reinicio esencialmente al azar. En otras palabras, se pretende estar haciendo un paseo en lugar de una búsqueda al azar. En segundo lugar, cuando el algoritmo descubre un nuevo óptimo local, decide si desea conservar el actual “punto de inicio” como óptimo local, o adoptar el nuevo óptimo local como el “punto de inicio”. Si siempre se escoge el nuevo óptimo local, se estaría haciendo un recorrido aleatorio (una especie de meta-exploración). Si siempre se elige el nuevo óptimo local con la condición que este sea mejor que el

actual, se estaría haciendo un Hill-Climbing (ascenso a la colina) simple (una especie de meta-explotación). A menudo se debe estar en algún lugar intermedio entre estos dos extremos [59].

2.9 ARREGLOS DE COBERTURA

Los arreglos de cobertura identificados como CA por sus siglas en inglés (Covering Arrays) son objetos combinatorios derivados de los arreglos ortogonales. Los CA se han utilizado exitosamente para automatizar la generación de casos de prueba para las pruebas de software. Estos arreglos tienen cardinalidad mínima (es decir, reducen al mínimo el número de casos de prueba) y cobertura máxima (garantizan cubrir todas las combinaciones de cierto tamaño entre los parámetros de entrada).

Un CA representado como $CA(N, t, k, v)$ puede ser visto también como una matriz de N filas y k columnas a lo largo de un vocabulario v de símbolos, de tal manera que para cada conjunto de columnas t (llamado la fuerza de la matriz) cada t-tupla (posible combinación de valores de parámetros) está cubierta por lo menos una vez [60]. El número de filas que define un CA indica el número de experimentos o pruebas que se realizarán, este tamaño depende de: la cantidad de columnas (cantidad de parámetros que interfieren en la prueba), el vocabulario escogido para la prueba (tamaño del vector de valores que se probarán para cada parámetro) y la fuerza escogida para la prueba (variable que indica la cantidad de columnas que se agrupan para la posterior combinación). En la Figura 2 se proporciona un ejemplo de un CA de fuerza dos ($t = 2$) con cuatro parámetros ($k = 4$) cada uno con tres posibles valores ($v = 3$).

0	0	0	0
0	1	1	1
0	2	2	2
1	0	1	2
1	1	2	0
1	2	0	1
2	0	2	1
2	1	0	2
2	2	1	0

Figura 2: Ejemplo $CA(9;\ 2, 4, 3)$.

Un diseño experimental completo o exhaustivo $(t = k)$ debe cubrir v^t experimentos, para el anterior ejemplo 3^4 = 81 experimentos, sin embargo, en este caso la interacción es relajada a dos ($t = 2$), entonces el número de combinaciones posibles se reduce a nueve (9) casos de prueba (que representa una reducción del 90 por ciento en el número de experimentos necesarios). Finalmente nótese que en cada par de columnas aparecen las combinaciones [(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2), (2, 0), (2, 1), (2, 2)] al menos una vez, lo que significa que es de fuerza 2.

En el presente trabajo, los CA se han usado para soportar la afinación de parámetros de los algoritmos y el análisis de sensibilidad de los mismos, ya que se

logran hacer pruebas con la menor cantidad de combinaciones posibles de valores, logrando obtener una cobertura máxima según el parámetro t previamente definido.

3. ESTADO DEL ARTE

3.1 INTRODUCCIÓN

En este apartado, se presentan las investigaciones más representativas relacionadas con la calibración de modelos de tráfico vehicular, con el objetivo de conocer las técnicas o procesos que se han empleado para calibrar modelos de tráfico vehicular; también se exponen investigaciones donde el proceso de calibración de diferentes modelos se ha realizado mediante algoritmos multi-objetivo; por último se presentan investigaciones donde se ha estudiado el proceso de optimización usando NSGA-II o MOGBHS y se muestran los resultados obtenidos por estos algoritmos.

3.2 CALIBRACIÓN DE MODELOS DE TRÁFICO VEHICULAR

Los modelos de micro-simulación se están convirtiendo en herramientas cada vez más importantes en el modelado de sistemas de transporte. Un gran número de modelos se utilizan en diversos países. La etapa más difícil en el desarrollo y uso de tales modelos es el proceso de calibración [61]. En este proceso la calidad de los parámetros del modelo final dependerá de la estructura del modelo, el poder del algoritmo de optimización, la calidad de los datos de entrada, y los criterios de estimación o funciones objetivo utilizados en el procedimiento de optimización [62]. Sin embargo, varias cuestiones metodológicas en la creación de un proceso de calibración para los modelos de micro-simulación de tráfico vehicular están aún sin resolver. La influencia de los parámetros individuales de los modelos microscópicos en la dinámica del tráfico simulado también está lejos de ser clara. Para hacer frente a este problema en [63] establecen una metodología basada en el análisis de sensibilidad de los modelos de flujo de tráfico. El análisis permitió identificar que los parámetros son susceptibles de ser calibrados de forma independiente, es decir, a partir de datos de diferentes lugares.

En la literatura se puede encontrar muchos trabajos relacionados con el tema de la calibración de modelos de tráfico vehicular. La mayoría de las investigaciones publicadas se centran en un solo componente del modelo de simulación, dejando a un lado los demás. Por ejemplo en [64] [65] [66] solo calibran los parámetros del comportamiento del conductor.

En [36] se presenta un framework para la calibración de los modelos de micro-simulación de tráfico vehicular. El marco de referencia utilizado tiene en cuenta las interacciones entre las diversas entradas y parámetros del simulador mediante la estimación de origen-destino. Un enfoque de búsqueda sistemática basada en el algoritmo Complex (algoritmo que contempla diversos métodos matemáticos) se adoptó para la calibración de los parámetros de comportamiento del modelo. Este algoritmo es particularmente útil para el problema en cuestión, ya que no requiere

cálculos derivados de la función objetivo, que es computacionalmente costoso en el contexto de un modelo de simulación estocástica. Además, el algoritmo usa un conjunto de puntos de partida iniciales repartidos al azar sobre el espacio de búsqueda y por lo tanto tiende a encontrar óptimos globales en lugar de locales.

Los algoritmos genéticos también han sido utilizados como herramienta de calibración para modelos de micro-simulación de tráfico vehicular. En la mayoría de los trabajos que los usan, se calibra sólo un subconjunto de parámetros. En [67] implementan una herramienta de optimización llamada GENOSIM desarrollada en Toronto, Canadá con el propósito de encontrar un proceso de calibración rápida, sistemática y robusta. Este software se compone de una serie de algoritmos genéticos que manipulan los valores de los parámetros de control y la búsqueda de un conjunto óptimo de valores que minimicen la discrepancia entre los datos de salida de la simulación y los de campo. Los resultados obtenidos mediante la replicación de conteos de vehículos observados fueron prometedores.

En [68] utilizan un algoritmo genético para buscar los valores óptimos de los parámetros de calibración de los modelos de rendimiento del vehículo utilizados por dos modelos de micro-simulación de tráfico vehicular CORSIM. La relación de error absoluto entre las curvas de rendimientos simulados y empíricos se utilizó como la función objetivo, estos datos empíricos fueron obtenidos a través de sensores instalados en diferentes camiones que circularon por numerosas carreteras. Los resultados obtenidos mostraron claramente la viabilidad del enfoque propuesto.

Un algoritmo genético también es utilizado en [69], en esta investigación recalcan la importancia del proceso de calibración para poder lograr fidelidad y credibilidad de un modelo de simulación de tráfico vehicular. Proponen un procedimiento para la calibración del modelo de simulación microscópica debido a que la mayoría de los esfuerzos de calibración descritos en la literatura se han centrado en la práctica informal, y rara vez se ha propuesto un procedimiento sistemático o directriz para la calibración y validación de modelos de simulación. La validez del procedimiento propuesto se demostró mediante el uso de un caso de estudio de una intersección señalizada mediante el uso de un modelo de simulación de tráfico microscópico VISSIM. Los resultados de la simulación se compararon con varios datos de campo tomados en diferentes días, para determinar el rendimiento del modelo calibrado. Se encontró que los parámetros calibrados obtenidos por el procedimiento propuesto generan medidas de rendimiento que eran representativas de las condiciones de campo.

Los algoritmos genéticos también han sido usados como un marco para comparar otros algoritmos que prometen ser una buena herramienta para la calibración de modelos de tráfico vehicular, un ejemplo de esto es [37] en donde se compara un algoritmo genético con otras tres técnicas: Búsqueda Tabú (TS), optimización por enjambre de partículas (PSO) y el algoritmo estocástico de perturbación simultanea (SPSA). Estas cuatro técnicas fueron utilizadas para calibrar un modelo de micro-simulación de tráfico llamado SUMO en dos diferentes casos de estudio. Los autores indican que cada técnica tiene sus propios parámetros que afectan la

velocidad de convergencia, por lo que recomienda que las técnicas de optimización sean previamente afinadas, pero este proceso consume tiempo y recursos. En la investigación se encontró que TS tiene un menor número de parámetros que deben ser afinados y este mismo no se vio afectado sensiblemente por ellos. También que PSO y TS obtienen mejores resultados que el algoritmo genético y el algoritmo SPSA. Además, que PSO es una técnica prometedora para la calibración de los modelos ya que éste puede ser paralelizado fácilmente, lo que reduce ampliamente su tiempo de ejecución.

Un algoritmo genético y una meta-heurística (Random Search) fue utilizado en [70] para calibrar los parámetros de un modelo, mostrando que la meta-heurística es más eficaz que el algoritmo genético, y que este último requiere un esfuerzo computacional adicional debido a un mayor número de evaluaciones de la función objetivo. Por tal razón en [71] se presentó una descripción detallada de los algoritmos meta-heurísticos más adecuados para el procedimiento de calibración de los modelos de flujo de tráfico vehicular, junto con sus principales características, ventajas y puntos débiles. Entre dichas metaheurísticas se presenta los algoritmos: método de entropía cruzada, recocido simulado, optimización por enjambre de partículas, Nelder–Mead. Este último fue utilizado como herramienta para la calibración de dos modelos de tráfico vehicular, demostrando que los modelos resultantes son capaces de reproducir las condiciones reales del tráfico con suficiente precisión.

En este sentido se encuentran diversas meta-heurísticas que se han utilizado para calibrar modelos de tráfico vehicular y que mejoran los resultados obtenidos por un algoritmo genético, tal es el caso del algoritmo de salto de rana que se presenta en [72] como una mejora al método de optimización estocástica, la propuesta mejoró los resultados entre 2.1% y 3.5%. Además, los autores señalan que esta propuesta resulta más eficiente que un algoritmo genético debido a su naturaleza de búsqueda, que combina la búsqueda local, así como búsqueda global, sin necesidad de evaluaciones adicionales que habrían sido necesarias en un algoritmo genético. El algoritmo de optimización por colonia de hormigas (ACO por sus siglas en inglés) es propuesto en [73]. El análisis estadístico mostró que ACO obtuvo mejores resultados en comparación con un algoritmo genético para los casos que utilizan un mayor número de ejecuciones del modelo, identificando mejores soluciones para la misma potencia de cálculo. Además los autores indican que este algoritmo puede ser una buena alternativa para la solución de redes más complejas. Por último, indican que el enfoque ACO demuestra el potencial de aplicación en una configuración de computación en paralelo, lo que podría reducir significativamente el tiempo de procesamiento para encontrar soluciones óptimas.

En [74] implementan el algoritmo OptQuest/Multistart como herramienta de calibración para modelos de micro-simulación de flujo de tráfico vehicular. El algoritmo demostró su capacidad para llegar a una solución óptima no muy lejos de la global. Por otra parte, los autores afirman que la búsqueda de una solución eficaz al problema de calibración no solo depende de la elección del algoritmo de

optimización más eficiente. El uso de la información disponible sobre el modelo podría permitir un mejor rendimiento en el proceso de calibración.

Un algoritmo memético denominado GASA es propuesto en [6] para la calibración de modelos de micro-simulación de flujo de tráfico. El algoritmo incluye una combinación entre un algoritmo genético y recocido simulado. El algoritmo genético lleva a cabo la exploración en todo el espacio de búsqueda e identifica una zona en la que se pudo localizar una posible solución global. Después de que se ha encontrado esta zona, el algoritmo de recocido simulado refina la búsqueda y localiza un conjunto óptimo de parámetros en esa zona. La metodología propuesta permite calibrar todos los parámetros del modelo, junto con múltiples medidas de rendimiento y períodos de tiempo de forma simultánea. Dos sistemas de tráfico vehicular CORSIM diferentes fueron calibrados para este estudio. Todos los parámetros después de la calibración se encontraban dentro de límites requeridos. En la investigación se proporciona una comparación entre GASA y el algoritmo SPSA, los resultados fueron similares entre los dos. El esfuerzo necesario para afinar a GASA fue considerablemente menor en comparación con el de SPSA. Sin embargo, el tiempo de ejecución del GASA fue mucho mayor en comparación con el tiempo de SPSA. Además, GASA requiere un cierto conocimiento del modelo con el fin de establecer los parámetros de optimización adecuados.

En [21] los autores proponen un algoritmo memético multi-objetivo basado en NSGA-II y Recocido Simulado (SA), denominado NSGA-II-SA para la calibración de modelos de micro-simulación de flujo de tráfico vehicular CORSIM. El algoritmo NSGA-II se encarga de realizar exploración en el espacio de búsqueda y obtiene el frente de Pareto el cual es optimizado completamente con SA. Dos modelos de tráfico vehicular CORSIM fueron calibrados con el algoritmo propuesto, el rendimiento de la propuesta memética es comparado con dos algoritmos del estado del arte: GASA y SPSA. Los resultados mostraron superioridad del algoritmo NSGA-II-SA en términos de tiempo de ejecución y convergencia.

Similarmente al anterior, en [75] proponen un algoritmo memético mono-objetivo basado en búsqueda local encadenada haciendo uso del algoritmo Solis & West SA para la calibración de modelos de micro-simulación de flujo de tráfico vehicular CORSIM. La propuesta fue evaluada sobre dos modelos de tráfico vehicular. Además, los algoritmos fueron comparados con dos técnicas del estado del arte, GASA y SPSA, los resultados indican que el algoritmo propuesto tiende a converger más rápido que las técnicas del estado del arte.

3.3 CALIBRACIÓN MEDIANTE ALGORITMOS MULTI-OBJETIVO

En la literatura se encuentran investigaciones que utilizan algoritmos multi-objetivo como instrumento de calibración, pero no para calibrar un modelo de flujo vehicular, la mayoría de estos se relacionan con modelos hidrológicos.

Una calibración automática de modelos hidrológicos de gran escala y de aplicaciones complejas utilizando el algoritmo multi-objetivo NSGA-II se presenta en

[76]. Y una evaluación global de la eficacia relativa de algoritmos multi-objetivo como herramientas de calibración de los modelos hidrológicos [77] muestra que ε-NSGA-II logra el mejor rendimiento global para el conjunto de prueba.

En [78] se presenta un algoritmo genético multi-objetivo para la calibración de un simulador de una planta de energía solar térmica, cuyos resultados sobresalen por su exactitud.

En [79] proponen un nuevo algoritmo multi-objetivo que combina un algoritmo genético multi-objetivo (MOGA) con redes neuronales adaptativas, para determinar las ubicaciones óptimas de muestreo para la instalación de registradores de presión en un sistema de distribución de agua, el propósito de la instalación de los registradores es recoger datos para la calibración de un modelo hidráulico. Los resultados muestran que el algoritmo propuesto requiere menos recursos computacionales en comparación con el MOGA y sin disminución significativa en la exactitud de la solución final.

3.4 OPTIMIZACIÓN USANDO NSGA-II Y MOGBHS

La mayoría de trabajos de investigación donde utilizan el algoritmo NSGA-II lo hacen para comparar resultados con nuevas propuestas. Otros modifican el algoritmo (de acuerdo al problema) agregando conocimiento específico del problema para obtener mejores resultados, disminuir los tiempos de ejecución o minimizar recursos computacionales; en estos casos, este algoritmo encuentra mejores soluciones que los algoritmos mono-objetivo que usan una función de ponderación que combina los diferentes objetivos [80].

En [80] se propone una nueva distancia de apiñamiento (crowding distance) dado que en el problema particular, la distancia de apiñamiento original descartaba puntos que se consideraban interesantes; con este cambio se logra obtener el verdadero frente de Pareto. Además, utilizan una estrategia de acercamiento de grueso a fino para refinar el modelo de aproximación y los resultados de optimización. Esto lo aplican a la optimización aerodinámica multi-objetivo del rotor 37 de la NASA, obteniendo mejores resultados (con un aumento leve en los costos computacionales) que la optimización mono-objetivo y otras soluciones alternativas.

En [81] buscaron mejorar la fiabilidad de las redes informáticas estocásticas (SCN-Stochastic Computer Network) y minimizar costos, para esto, usaron el algoritmo NSGA-II y el algoritmo SPGA-II (Sub-Population Genetic Algorithm II), para encontrar los frentes de Pareto. Lograron determinar que el algoritmo NSGA-II tiene una mejor eficiencia computacional en la búsqueda de las soluciones de Pareto que el algoritmo SPGA-II.

En [46] proponen una nueva estrategia de control multi-objetivo para optimizar al mismo tiempo la demanda de efluentes químicas de oxígeno (COD_{eff}) y el caudal de biogás (Q_{gas}) en un bio-reactor anaeróbico, usando el algoritmo NSGA-II y el algoritmo de red neuronal artificial genético (GA-ANN). El algoritmo NSGA-II

optimiza las variables del modelo de la red neuronal con el objetivo de obtener una mayor precisión y lograr alcanzar mejores resultados. Los resultados obtenidos demuestran que la propuesta es eficiente para el problema específico.

El algoritmo MOGBHS es un nuevo algoritmo meta-heurístico, el cual utiliza los conceptos de ordenamiento de soluciones no dominadas (para realizar optimización multi-objetivo) y la mejor búsqueda armónica global (Global-best Harmony Search), En [20] lo prueban en un modelo de simulación de un sistema de transporte masivo de pasajeros con el objetivo de establecer las frecuencias de salida de cada ruta del sistema de transporte, con el fin de minimizar los costos y maximizar la calidad del servicio. El algoritmo propuesto se compara con el NSGA-II, demostrando que para el problema en particular, éste es más eficaz y obtiene mejores soluciones del frente de Pareto. También ha sido usado para la definición de rutas y horarios en sistemas de transporte masivo de pasajeros obteniendo resultados prometedores, pero esta información todavía está en proceso de publicación.

CAPÍTULO 3

4. PROPUESTA

4.1 INTRODUCCIÓN

En la presente sección se muestran las funciones objetivo utilizadas para la presente investigación y el criterio con el que se determina si un modelo se encuentra calibrado. También, la representación de un individuo o solución y lo que significa que una solución sea considerada como una mejor solución del problema. Además, se muestran en detalle los algoritmos de búsqueda local (algoritmos de explotación) que fueron aplicados al problema de calibración y los algoritmos de búsqueda global que se utilizaron en el proyecto de investigación y la manera en cómo se convirtieron estos en una propuesta memética.

4.2 FUNCIONES OBJETIVO Y CRITERIO DE CALIBRACIÓN

Al trabajar esta propuesta en un marco multi-objetivo se hace necesario manejar diferentes funciones objetivo. En este estudio se utilizaron dos objetivos: volumen y velocidad, expresados matemáticamente con las ecuaciones (1) y (2) respectivamente. Estos objetivos fueron seleccionados debido a su importancia en este problema como se presentó en la sección 1.1 (Planteamiento del problema). Las fórmulas planteadas para cada objetivo son una adaptación de la función de error propuesta en [6], en la cual agrupan los dos objetivos en una sola función (ver ecuación (3)). Esta fórmula también fue utilizada en este estudio para obtener el mejor individuo del frente de Pareto y realizar una comparación con los algoritmos mono-objetivo (ver sección 5).

$$Volumen = \frac{1}{\sqrt{N}} \sum_{t=1}^{T} \sqrt{\sum_{i=1}^{N} \left(\frac{V_{i,t} - \tilde{V}_{(\theta)i,t}}{V_{i,t}}\right)^2} \qquad (1)$$

Donde $V_{i,t}$ es el conteo de enlaces reales para el enlace i en el tiempo t, $\tilde{V}_{(\theta)i,t}$ es el conteo de enlaces simulados para el enlace i en el tiempo t, N es el número total de enlaces en el modelo y T es el número total de períodos de tiempo t.

$$Velocidad = \frac{1}{\sqrt{N}} \sum_{t=1}^{T} \sqrt{\sum_{i=1}^{N} \left(\frac{S_{i,t} - \tilde{S}_{(\theta)i,t}}{S_{i,t}}\right)^2} \qquad (2)$$

Donde $S_{i,t}$ es la velocidad real para el enlace i en el tiempo t, $\tilde{S}_{(\theta)i,t}$ es la velocidad simulada para el enlace i en el tiempo t, N es el número total de enlaces en el modelo y T es el número total de períodos de tiempo t.

$$NRMS = \mathrm{W} * \mathrm{Volumen} + (1 - \mathrm{W}) * \mathrm{Velocidad} \quad (3)$$

Donde W es un peso utilizado para asignarle mayor o menor ponderación al reconteo (volumen) y velocidad.

Criterio de calibración: En la presente investigación se utilizaron las directrices de la Administración Federal de Carreteras (FHWA) para los modelos CORSIM. La diferencia entre el conteo de enlaces reales y simulados debe ser inferior a 5% para todos los enlaces. Por esto se usó el estadístico GEH [82], el cual debe ser inferior a 5 por lo menos en 85% de los enlaces. La fórmula utilizada para calcular el estadístico GEH fue la misma que utilizan en [6] la cual es expuesta en la ecuación (4).

$$GEH = \sqrt{\frac{2(V_i - \tilde{V}_{(\theta)i})^2}{V_i + \tilde{V}_{(\theta)i}}} \quad (4)$$

Donde V_i es el conteo de enlaces reales para el enlace i y $\tilde{V}_{(\theta)i}$ es el conteo de enlaces simulados para el enlace i.

4.3 REPRESENTACIÓN DE UN INDIVIDUO

Un individuo se representa como un objeto el cual varía en un tiempo t (t, iteración actual del algoritmo) y está compuesto por lo siguiente:

- Vector que representa los valores de los parámetros del modelo a ser simulado.
- Valor de aptitud para el objetivo de velocidad.
- Valor de aptitud para el objetivo de volumen.
- Valor de aptitud para el error cuadrático medio normalizado (NRMS).
- Valor del ranking del individuo, el cual indica en que frente de Pareto se encuentra el individuo.
- Valor que indica cuantos individuos dominan al individuo actual.
- Valor que indica que tan dispersa es la solución dentro del frente al que pertenece (distancia de crowding).
- Lista de individuos a los cuales domina el individuo actual.

En un tiempo inicial ($t = 0$) el individuo solo cuenta con la información de los parámetros iniciales del vector generados aleatoriamente en el momento que se inicializa la población. Posterior a esto (en un tiempo $t = 1$), los valores iniciales son evaluados con las funciones objetivo y así se obtienen los tres valores de aptitud (estos valores de aptitud se obtienen por medio de la simulación del modelo generado a partir de los valores del vector y la posterior comparación con los datos

de campo). Finalmente (en un tiempo $t = i$), los valores de velocidad y volumen del individuo son comparados con los demás individuos de la población para: obtener el frente al que pertenece (ranking) en el caso de los algoritmos NSGA-II y MOGBHS, los individuos que son dominados por él, la cantidad individuos que dominan al individuo actual y la asignación de su aptitud en el caso del algoritmo SPEA-2. Finalmente se calcula la distancia de apiñamiento de dicho individuo. El tiempo $t = i$ dependerá del algoritmo de exploración seleccionado (esto se detalla más adelante en las secciones 4.9, 4.10, 4.11).

El vector que representa el modelo a ser simulado contiene, entre otros, el conjunto de valores que representan la solución para el problema de calibración. Cada parámetro tiene un conjunto de valores posibles o válidos que está dado por CORSIM. La Figura 3 muestra la estructura y contenido de los archivos planos que utiliza el simulador CORSIM como entrada para el proceso de simulación. En este archivo cada línea representa un record type (RT), la información del mismo se encuentra agrupada en conjuntos de cuatro columnas las cuales son denominadas entradas (entries).

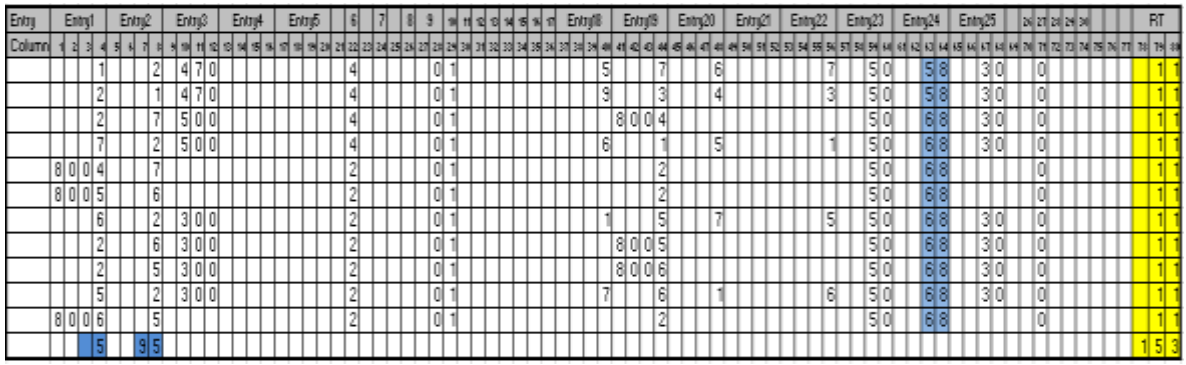

Figura 3. Estructura del archivo plano en formato de CORSIM.

En la anterior figura se ilustra los record type 11 y 153 resaltados en amarillo y en azul claro las entradas que los record type están recibiendo. Para el caso del record type 11 que contiene información sobre las características de una calle se están tomando los valores de la entrada 24 que representa el promedio de avance de descarga de colas. El record type 153 contiene la información acerca de la familiaridad del conductor con las rutas de distribución, para este caso se tiene la información referente a la entrada 1 que contiene el porcentaje de conductores que conocen sólo un movimiento de giro y la entrada 2 que contiene el porcentaje de conductores que conocen dos movimientos de giro. Los valores resaltados en azul claro son los datos a calibrar.

Una vez obtenidos los datos que se deben calibrar, la representación de la solución resultante se muestra en la Figura 4.

Figura 4. Ejemplo de la representación de la solución (vector de parámetros).

Con lo anterior se deduce que el tamaño de la solución depende de la cantidad de record type con sus respectivas entradas que sean seleccionadas en el modelo que se desea calibrar. Los valores que puede tomar cada parámetro deben ser enteros, además de esto, cada uno de ellos deben estar en un determinado rango y estos deben conservar coherencia, tal es el ejemplo del record type 153 donde se toman dos entradas en las cuales los valores representan porcentajes, con ello se tiene que, la suma de estos dos valores debe ser 100 y estos no pueden tomar valores inferiores a 0 ni superiores a 100.

Para tener una explicación más detallada del significado de cada uno de los record type con sus respectivas entradas y restricciones se recomienda ver el manual de referencia CORSIM el cual se presenta como anexo del presente documento.

4.4 DESCRIPCIÓN DE UNA SOLUCIÓN VECINA

Una solución vecina parte de una solución actual, a la cual se le aplica algún cambio en algunos de sus parámetros con el objetivo de encontrar una solución óptima para el problema de optimización. El cambio que se aplica al vector solución de un individuo depende del algoritmo utilizado. A continuación, se describe como dichos algoritmos generan una solución vecina.

4.4.1 SOLUCIÓN VECINA GENERADA A PARTIR DEL ALGORITMO NSGA-II

El NSGA-II tiene sus bases en un algoritmo genético, por lo tanto este no solo genera una sola solución en cada iteración sino una población de descendientes, la forma en como esto se realiza es a través de métodos de selección, cruce y mutación; los cuales operan sobre individuos pertenecientes a una población. Se selecciona una cantidad n de parejas de individuos (padres) de una población de tamaño N, posterior a esto se forman parejas de padres para cruzarlos entre sí, con ello se obtienen una cantidad $2n$ de hijos. Finalmente, a cada individuo perteneciente a esta última población (hijos) se le aplica una mutación, lo cual altera una cierta cantidad de parámetros del vector que representa la solución.

A continuación, se describe los métodos de selección, cruce y mutación con los cuales se obtuvo mejores resultados.

4.4.1.1 MÉTODO DE SELECCIÓN: ELITISMO

La población es ordenada de forma ascendente a partir de los atributos de: número de Rank (frente al que pertenece) y distancia de Crowding, luego de esto una cantidad determinada de soluciones (para esta investigación esta cantidad es $2n$) es seleccionada para posteriores operaciones. Para la presente investigación se cuenta también con la implementación de los métodos de selección: torneo y ruleta.

4.4.1.2 MÉTODO DE CRUCE: UNIFORME

Partiendo de dos individuos (padre y madre respectivamente), se genera una "máscara" (vector binario) aleatoria de igual tamaño del vector solución que contiene

el individuo. Si en la posición i de la máscara el valor es 0 la posición i del vector del primer hijo toma el valor de la posición i del vector de la madre y la posición i del segundo hijo toma el valor de la posición i del vector del padre. Si en la posición i de la máscara el valor es 1 la posición i del vector del primer hijo toma el valor de la posición i del vector del padre y la posición i del segundo hijo toma el valor de la posición i del vector de la madre. En ésta investigación también se implementaron los métodos de cruce de un punto y de dos puntos.

4.4.1.3 MÉTODO DE MUTACIÓN: MULTIGEN

Todos los hijos resultantes son mutados, alterando una cantidad de genes del vector solución, esta cantidad depende de un porcentaje de mutación definido por el usuario, en cada posición del vector solución se evalúa una probabilidad de que se realice o no una alteración del gen. La alteración consiste en cambiar el gen (valor en la posición i) por uno generado aleatoriamente teniendo en cuenta las restricciones de los valores que puede tomar esa posición. En ésta investigación también se implementó el método de mutación de un solo gen.

4.4.2 SOLUCIÓN VECINA GENERADA A PARTIR DEL ALGORITMO MOGBHS

El algoritmo MOGBHS parte del algoritmo de la mejor búsqueda armónica global, por lo tanto, un individuo es generado en cada iteración de diversas soluciones e incluso, algunos parámetros pueden ser generados de valores aleatorios. En este caso se modificó la generación de un nuevo individuo, este procedimiento se explicará en la sección 4.13.

El parámetro de porcentaje de consideración de la memoria armónica (HMCR) fija si el valor en la posición i del vector solución del individuo, es obtenido desde la memoria armónica o generado de forma aleatoria (cumpliendo con las restricciones). Si es obtenido desde la memoria armónica, el parámetro de porcentaje de ajuste de tono (PAR) define si el valor es obtenido desde el mejor individuo de la memoria armónica (en este caso, un individuo al azar del frente de Pareto), en caso contrario, se obtiene desde un individuo al azar de la memoria armónica.

El parámetro PAR es derivado a partir de los parámetros: porcentaje máximo de ajuste de tono (PARmax), porcentaje mínimo de ajuste de tono (PARmin), número de improvisaciones y la improvisación actual.

4.4.3 SOLUCIÓN VECINA GENERADA A PARTIR DEL ALGORITMO SPEA-2

Al igual que NSGA-II, este algoritmo es un enfoque bio-inspirado, que tiene sus bases en un algoritmo genético, por lo tanto, este no solo genera una sola solución en cada iteración sino una población de descendiente, la manera en como esta población se genera es mediante métodos inspirados en la evolución biológica, la selección de los padres utilizados para la posterior reproducción se hace a través de la población externa que contiene las mejores soluciones. El método de selección recomendado en la literatura es el de torneo. Los métodos de cruce y mutación son

los mismos métodos empleados en NSGA-II (uniforme y multi-gen respectivamente).

4.4.4 SOLUCIÓN VECINA GENERADA A PARTIR DE LOS ALGORITMOS DE BÚSQUEDA LOCAL

Se cuenta con tres algoritmos de búsqueda local: ascenso a la colina, recocido simulado y búsqueda local iterada, este último utiliza dentro de él un algoritmo de explotación. Los algoritmos ascenso a la colina y recocido simulado generan un individuo vecino a partir de un individuo actual aplicando una perturbación a una cantidad determinada de parámetros del vector solución, una vez se decide si un parámetro va a ser perturbado hay una probabilidad del cincuenta por ciento (50%) que determina si a ese valor se le sumará o restará una porción del rango. En recocido simulado esa porción equivale a un porcentaje determinado, mientras que en ascenso a la colina equivale al valor del radio para problemas continuos.

El algoritmo de búsqueda local iterada genera un individuo vecino a partir del individuo actual que ha sido optimizado con un método de búsqueda local (para este proyecto, algoritmo ascenso a la colina), al vector solución del individuo actual se le realiza una perturbación a una cantidad determinada de sus parámetros, dicha cantidad se establece en un valor el cual no es ni muy pequeño para impedir que el nuevo individuo se mantenga en la misma zona, ni muy alto para evitar una búsqueda al azar. La perturbación consiste en generar valores aleatorios dentro del rango de los parámetros y reemplazarlos en las posiciones correspondientes del vector solución del vecino generado a partir del algoritmo de búsqueda local, obteniendo así al nuevo óptimo local.

4.5 DESCRIPCIÓN DE UNA MEJOR SOLUCIÓN

Una solución candidata es aquella que al evaluarla en una función objetivo, mejora su valor de aptitud en comparación con la solución actual. Al tratarse esta investigación en un marco multi-objetivo se hace uso del concepto de dominancia para determinar si una solución es mejor que otra, dicho concepto especifica que para que una solución domine a otra (sea mejor) esta debe tener mejor valor de aptitud en al menos un objetivo y, por lo menos, igualar el valor de aptitud en los demás objetivos.

Más adelante se muestra, a manera de ejemplo, cómo un algoritmo de búsqueda local genera una solución vecina y cómo se decide si esta es una mejor solución al problema.

4.6 ALGORITMO DE BÚSQUEDA LOCAL: ASCENSO A LA COLINA

La Figura 5 ilustra el Pseudocódigo del algoritmo ascenso a la colina utilizado en este trabajo. Este algoritmo repite los procedimientos de las líneas 3 a la 7 mientras que el criterio de parada no se cumpla. En la presente investigación el criterio de parada se cumple cuando el algoritmo realiza una cantidad determinada de

iteraciones (igual para los demás algoritmos de búsqueda local y global). La función obtener mejor vecino tiene el objetivo de generar una cantidad n de vecinos cercanos de la solución actual, la generación de un vecino se realiza mediante una perturbación del vector solución del individuo actual (haciendo uso de la función que se explicó anteriormente en la sección 4.4.4), posterior a esto calcula las aptitudes a partir del vector solución de los vecinos generados y se obtiene el mejor de ellos, finalmente si este vecino domina a la solución actual, este será considerado como una mejor solución y pasará a ser la nueva solución actual ($Best$).

Pseudocódigo Algoritmo Búsqueda Local: Ascenso a la colina

Entrada: Individuo inicial s; Radio r; Número de iteraciones N; Porcentaje perturbar p
Salida: Individuo optimizado $Best$

1: $Best =$ copia(s), $i = 0$
2: **Mientras** no se cumpla criterio de parada ($i < N$) **hacer**
3: $R =$ obtener mejor vecino($r, p, Best$)
4: **Si** R domina a $Best$ **entonces**
5: $Best = R$
6: **Fin si**
7: incrementar i en 1
8: **Fin mientras**
9: **Retornar** $Best$

Figura 5: Pseudocódigo algoritmo ascenso a la colina.

A continuación, se ilustra cómo el algoritmo ascenso a la colina genera una solución vecina y como decide si esta es una mejor solución para el problema.

Se parte de una solución inicial S_i como la siguiente:

X_1	X_2	X_3	X_4	…….	X_n	$Aptitud\ Volumen$	$Aptitud\ Velocidad$

Una cantidad determinada de parámetros de la solución inicial es perturbada con el mecanismo mencionado anteriormente, la perturbación a estos parámetros equivale a un porcentaje del rango del mismo (P_i), con una probabilidad del 50% para establecer si las alteraciones son positivas o negativas. Con esto se genera una solución vecina S_i^*.

$X_1 + P_1$	X_2	$X_3 - P_3$	X_4	…	X_n	$Aptitud\ Volumen^*$	$Aptitud\ Velocidad^*$

Posterior a esto, se evalúa si la solución vecina generada domina a la solución actual con el objetivo de verificar si la nueva solución es considerada una mejor solución para el problema. La Figura 6 muestra el pseudocódigo del método utilizado para establecer si una solución domina a otra.

Pseudocódigo Método: Domina A

Entrada: Individuo actual S_i; Individuo vecino S_i^*
Salida: booleano (indica si S_i domina a S_i^*)

1: **Si** Volumen (S_i) > Volumen (S_i^*) o Velocidad (S_i) > Velocidad (S_i^*) **entonces**
2: **Retornar** falso
3: **Fin si**
4: **Si** Velocidad (S_i) < Velocidad (S_i^*) o Volumen (S_i) < Volumen (S_i^*) **entonces**
5: **Retornar** verdadero
6: **Fin si**
7: **Retornar** falso (En caso de tener valores de aptitud iguales, no se puede determinar dominancia)

Figura 6: Pseudocódigo Método Domina A.

Finalmente, si la solución vecina cumple con el criterio de dominancia esta pasará a ser la nueva solución actual, de no establecerse una dominancia, una probabilidad del 50% determina cuál de las dos soluciones es tomada como la solución actual.

4.7 ALGORITMO DE BÚSQUEDA LOCAL: RECOCIDO SIMULADO

La Figura 7 ilustra el Pseudocódigo del algoritmo recocido simulado utilizado. Este algoritmo repite los procedimientos de las líneas 3 a la 11 mientras que el criterio de parada no se cumpla. En la línea 3 se genera una cantidad n de vecinos cercanos de la solución actual, usando el concepto anteriormente definido. Posterior a esto calcula las aptitudes a partir del vector solución de los vecinos generados y se obtiene el mejor de ellos. Luego en la línea 4 se determina si la nueva solución domina a la solución actual haciendo uso del concepto de dominancia (también definido previamente), de ser así, esta pasa a ser la solución actual ($Best$), de lo contrario se evalúa la probabilidad de aceptación, la cual está dada por la ecuación (5).

$$e^{-\frac{\left|Volumen_{(S_i)} - Volumen_{(S_i^*)}\right| + \left|Velocidad_{(S_i)} - Velocidad_{(S_i^*)}\right|}{2*T}} \qquad (5)$$

Esta probabilidad retorna un valor entre 0 y 1, la nueva solución es aceptada como la solución actual del algoritmo siempre y cuando el valor retornado por la probabilidad de aceptación sea menor que la constante de aceptación $\propto$. Finalmente se calcula el valor de la nueva temperatura mediante una función geométrica que consiste en multiplicar la temperatura actual por una tasa de enfriamiento σ.

Pseudocódigo Algoritmo Búsqueda Local: Recocido Simulado

Entrada: Individuo inicial s; Temperatura T; Tasa de enfriamiento σ; Porcentaje cambio δ; Constante aceptación α; Número de iteraciones N
Salida: Individuo optimizado $Best$

1: $Best =$ copia(s), $i = 0$
2: **Mientras** no se cumpla criterio de parada ($i < N$) **hacer**
3: $Next =$ obtener mejor vecino($Best, \delta$)
4: **Si** $Best$ domina a $Next$ **entonces**
5: **Si** probabilidad aceptación ($Best, Next, T$) <= α **entonces**
6: $Best = Next$
7: **Fin si**
8: **Si no**
9: $Best = Next$
10: **Fin si**
11: incrementar i en 1, $T = \sigma * T$
12: **Fin mientras**
13: **Retornar** $Best$

Figura 7: Pseudocódigo algoritmo recocido simulado.

4.8 ALGORITMO DE BÚSQUEDA LOCAL: BÚSQUEDA LOCAL ITERADA

La Figura 8 ilustra el Pseudocódigo del algoritmo búsqueda local iterada utilizado. En la línea 1 el algoritmo genera un número aleatorio entre uno (1) y el número de iteraciones que recibe como parámetro, definiendo así la cantidad de veces que itere el método de búsqueda local para encontrar un óptimo local. Luego se repiten los procedimientos de las líneas 4 a la 13 mientras que el criterio de parada no sea satisfecho. Con el propósito de pasar de un óptimo local a otro que sea cercano dentro del subespacio de soluciones, en la línea 4 se genera una cantidad n de vecinos cercanos de la solución actual, haciendo una perturbación al vector solución del óptimo local obtenido anteriormente. Posterior a esto calcula las aptitudes a partir del vector solución de los vecinos generados y se obtiene el mejor de ellos, generando así el un nuevo individuo $Next$. Posteriormente, se aplica el mismo método de búsqueda local al nuevo individuo, la cantidad de iteraciones para este método resultará de la generación de un número aleatorio entre uno (1) y la cantidad restante de iteraciones iniciales. Luego de esto las aptitudes del nuevo individuo son calculadas a partir de su vector solución. Finalmente, al igual que en el algoritmo de recocido simulado una probabilidad de aceptación es aplicada para determinar si la nueva solución pasa a ser la solución actual del algoritmo ($Best$). La fórmula para esto es similar a la utilizada por la ecuación (5) con la diferencia de que, en lugar de una temperatura variable, esta recibe como dividendo una probabilidad de aceptación σ que se mantiene constante a lo largo de las iteraciones del algoritmo.

Pseudocódigo Algoritmo Búsqueda Local: Búsqueda Local Iterada

Entrada: Individuo inicial s; Porcentaje perturbar δ; Probabilidad aceptación σ; Constante de aceptación α; Número de iteraciones N
Salida: Individuo optimizado $Best$
1: $k =$ Aleatorio entre (1, N), $i = k$

2: $Best = Busqueda\ local(s, k)$
3: **Mientras** no se cumpla criterio de parada ($i < N$) **hacer**
4: $Next =$ obtener mejor vecino($Best, \delta$)
5: $k =$ aleatorio entre (1, $(N - i)$), $i = i + k$
6: $Next =$ búsqueda local($Next, k$)
7: **Si** $Best$ domina a $Next$ **entonces**
8: **Si** probabilidad aceptación ($Best, Next, \sigma$) <= α **entonces**
9: $Best = Next$
10: **Fin si**
11: **Si no**
12: $Best = Next$
13: **Fin si**
14: **Fin mientras**
15: **Retornar** $Best$

Figura 8: Pseudocódigo algoritmo búsqueda local iterada.

4.9 ALGORITMO GENÉTICO BASADO EN ORDENAMIENTO NO DOMINADO-II (NSGA-II)

La Figura 9 ilustra el Pseudocódigo del algoritmo NSGA-II utilizado. Como ocurre en los algoritmos genéticos, en un tiempo $t = 0$, la primera población de individuos P_0 se obtiene aleatoriamente; a partir de ella se obtiene la primera población de descendientes, denotada por Q_0, de tamaño N. Luego de esto en la línea 3 de la Figura 9 se agrupan las poblaciones de padres y descendientes, P_t y Q_t, en una población global llamada R_t. Utilizando el concepto de frentes o ranking de no-dominancia se agrupan los diferentes individuos en dichos frentes: F_0 para los individuos que forman el frente de Pareto (ranking=0), F_1 para aquellos con ranking=1, etc. Para generar los frentes, en la línea 5 es utilizado el método de ordenamiento rápido por no dominancia. Posteriormente en la línea 14 se crea la siguiente población P_{t+1}. En esta nueva población se incluyen los individuos de F, frente por frente, empezando por el frente con ranking igual a 0, hasta que se hallan llenado los N elementos en la nueva población o un número cercano a N. Si no se ha logrado completar totalmente la nueva población P_{t+1} con frentes exactos, se debe seleccionar los $N - |P_{t+1}|$ individuos necesarios del frente F_i. Para ello se ordena el frente utilizando la distancia de apiñamiento (calculada en la línea 13), con el objetivo de incluir los individuos más diversos. Luego en la línea 19 se obtiene la población de descendientes Q_t aplicando el método de reproducción a la población P_{t+1} ilustrado en la Figura 10, dicho método utiliza los operadores de selección, cruce y mutación previamente descritos. Por último, se avanza una generación $t = t + 1$.

Este algoritmo repite los procedimientos de las líneas 3 a la 20 mientras que el criterio de parada no se cumple. Los procedimientos descritos entre las líneas 6 y 10 no hacen parte de la versión básica de NSGA-II dado que éste no recibe ningún

algoritmo de búsqueda local como parámetro para la variable de nombre $local$. Estos procedimientos solo se tienen en cuenta en el enfoque memético multi-objetivo descrito más adelante.

Pseudocódigo Algoritmo Multi-objetivo: NSGA-II

Entrada: Población inicial P_0; Número de iteraciones T; Tamaño población N; Algoritmo de búsqueda local $local$; Probabilidad de realizar explotación r
Salida: Población final P_T

1: $t = 0$
2: **Mientras** no se cumpla criterio de parada ($t < T$) **hacer**
3: $R_t = P_t \cup Q_{t-1}$
4: evaluar objetivos(R_t)
5: $F =$ ordenamiento rápido de no dominancia (R_t)
6: **Si** $local \neq \emptyset$ **entonces**
7: **Si** aleatorio entre $(0,1) < r$ **entonces**
8: búsqueda local(F_0)
9: **Fin si**
10: **Fin si**
11: $i = 0,\ P_{t+1} = \emptyset$
12: **Mientras** $|P_{t+1}| + |F_i| < N$ **hacer**
13: cálculo distancia de apiñamiento (F_i)
14: $P_{t+1} = P_{t+1} \cup F_i$
15: incrementar i en 1
16: **Fin mientras**
17: ordenar (F_i)
18: $P_{t+1} = P_{t+1} \cup F_i\ [1 : (N - |P_{t+1}|)]$
19: $Qt =$ reproducir (P_{t+1})
20: incrementar t en 1
21: **Fin mientras**
22: **Retornar** $\boldsymbol{P_T}$

Figura 9: Pseudocódigo algoritmo NSGA-II.

Para tener una descripción detallada de los métodos de ordenamiento rápido por no dominancia y distancia de apiñamiento se recomienda ver [14].

Pseudocódigo Método: Reproducir

Entrada: Población P; Cantidad de parejas n
Salida: Población de descendientes Q

1: $cantidad = 2 * n$, $Q = \emptyset$

2: $seleccionados$ = seleccionar ($P, cantidad$)
3: **Para** $i = 0$ **hasta** $cantidad$ **incremento** en 2 **hacer**
4: $padre = seleccionados_i$, $madre = seleccionados_{i+1}$
5: $hijos$ = cruzar ($padre, madre$)
6: **Para** $j = 0$ **hasta** $|hijos|$ **incremento** en 1 **hacer**
7: $Q = Q \cup$ mutar ($hijos_j$)
8: **Fin para**
9: **Fin para**
10: **Retornar** Q

Figura 10: Pseudocódigo Método Reproducir.

4.10 MEJOR BÚSQUEDA GLOBAL ARMÓNICA MULTI-OBJETIVO (MOGBHS)

La Figura 11 ilustra el pseudocódigo de la adaptación que se realizó en este trabajo al algoritmo MOGBHS [20]. Los individuos de la memoria armónica (HM) inicial son generados de forma aleatoria. Las aptitudes de todos los individuos son calculadas a partir de su vector solución, los individuos pertenecientes a la HM son ordenados a partir del frente y la distancia de apiñamiento. El algoritmo repite los procedimientos de las líneas 7 a la 30 mientras que el criterio de parada no sea satisfecho. En cada iteración del algoritmo, un nuevo individuo (Armonía) es generado de acuerdo a la descripción realizada previamente (ver sección 4.4.2). Esta nueva armonía es agregada a la HM (línea 20) y se procede nuevamente a evaluar y ordenar la nueva HM (líneas 21-22-23). Después de esto en la línea 24, se elimina la peor armonía de la memoria, es decir se elimina la última posición de la HM. Finalmente se incrementa el contador de generaciones $t = t + 1$.

Los procedimientos descritos entre las líneas 25 y 29 no son tomados en cuenta en la propuesta básica de MOGBHS dado que esta no recibe ningún algoritmo de búsqueda local como parámetro para la variable $local$. Estos procedimientos solo se tienen en cuenta en el enfoque memético multi-objetivo descrito más adelante.

El cálculo del PAR está dado por la ecuación (6).

$$PAR = PARMin + (((PARMax - PARMin) * t) / T) \quad (6)$$

Pseudocódigo Algoritmo Multi-objetivo: MOGBHS

Entrada: Memoria armónica HM; Número de iteraciones T; Tamaño población N; $HMCR$; $PARmin$; $PARmax$; dimensión del vector solución n; Algoritmo de búsqueda local $local$; Probabilidad de realizar explotación r
Salida: Última memoria armónica HM
1: $t = 0$
2: evaluar objetivos(HM)

3: ordenamiento rápido de no dominancia(HM)
4: cálculo distancia de apiñamiento(HM)
5: ordenar (HM)
6: **Mientras** no se cumpla criterio de parada ($t < T$) **hacer**
7: nuevaArmonía = $\emptyset$
8: **Para** $i = 0$ **hasta** n **incremento** en 1 **hacer**
9: **Si** aleatorio entre (0, 1) < $HMCR$ **entonces**
10: individuo = obtener individuo al azar de la HM
11 PAR = calcular PAR ($PARmin, PARmax, t$)
12: **Si** aleatorio entre (0, 1) < PAR **entonces**
13: individuo = obtener individuo al azar del frente de Pareto
14: **Fin si**
15: nuevaArmonía.vectorSolucion[i] = individuo.vectorSolucion[i]
16: **Si no**
17: nuevaArmonía.vectorSolucion[i] = númeroAleatorio(rango) //Se tiene en cuenta las restricciones para ese parámetro
18: **Fin si**
19: **Fin para**
20: $HM = HM \cup$ nuevaArmonia
21 ordenamiento rápido de no dominancia(HM)
22: cálculo distancia de apiñamiento(HM)
23: ordenar (HM)
24: eliminar último individuo (HM)
25: **Si** $local \neq \emptyset$ **entonces**
26: **Si** aleatorio entre $(0,1) < r$ **entonces**
27: búsqueda local(HM)
28: **Fin si**
29: **Fin si**
30: Incrementar t en 1
31: **Fin mientras**
32: **Retornar** $\boldsymbol{HM}$

Figura 11: Pseudocódigo algoritmo MOGBHS.

4.11 ALGORITMO MULTI-OBJETIVO DE FUERZA DE PARETO-2 (SPEA-2)

El algoritmo SPEA-2 utilizado en la presente investigación es una adaptación del mismo algoritmo encontrado en el framework propuesto en [83]. La Figura 12 ilustra el pseudocódigo del algoritmo.

El algoritmo empieza con una población inicial generada de forma aleatoria y la población externa vacía, luego el algoritmo itera entre las líneas 3 y 13 mientras el

criterio de parada no sea satisfecho. Las poblaciones general y externa (P_t y $\bar{P}_t$ respectivamente) son agrupadas en una población global R_t, los valores de aptitud de los objetivos de estudio son calculados a partir del vector solución de los individuos pertenecientes a la población R_t. Posteriormente en la línea 5 la aptitud (utilizada para verificar si el individuo es una solución óptima del problema, es decir, es parte del frente de Pareto) de todos los individuos de R_t es calculada, esta aptitud tiene en cuenta cuántos individuos domina cada individuo y cuántos lo dominan a él, además de la distancia con el vecino más cercano. Una selección de entorno (Figura 13) es realizada luego con el objetivo de copiar todos los individuos no-dominados, o en su defecto los de mejor aptitud, de la población Rt en la siguiente generación de la población externa $\bar{P}_{t+1}$. En este procedimiento se puede presentar tres posibilidades:

- La cantidad de individuos no-dominados de R_t sea igual al tamaño de la población externa $\bar{N}$, en este caso todas las soluciones son copiadas sin ningún problema.
- Si se excede el tamaño de $\bar{N}$, un operador de truncamiento es utilizado para eliminar las soluciones no-dominadas que se encuentren más apiñadas.
- Por último, si la cantidad de soluciones no-dominadas no completa el tamaño de la población externa esta se completa con los mejores individuos dominados de R_t.

En la línea 7 se reproducen los individuos obtenidos por la selección de entorno, los descendientes harán parte de la siguiente generación de la población general P_{t+1}. Por último, se avanza una generación $t = t + 1$.

Los procedimientos descritos entre las líneas 8 y 12 no son tomados en cuenta en la propuesta básica de SPEA-2 dado que esta no recibe ningún algoritmo de búsqueda local como parámetro para la variable $local$. Estos procedimientos solo se tienen en cuenta en el enfoque memético multi-objetivo descrito más adelante.

Pseudocódigo Algoritmo Multi-objetivo: SPEA-2

Entrada: Población inicial P_0; Número de iteraciones T; Tamaño población N; Tamaño población externa $\bar{N}$; Algoritmo de búsqueda local $local$; Probabilidad de realizar explotación r
Salida: Población externa final $\bar{P}_N$

1: $t = 0, \overline{P_0} = \emptyset$
2: **Mientras** no se cumpla criterio de parada ($t < T$) **hacer**
3: $R_t = Pt \cup \bar{P}_t$
4: evaluar objetivos(R_t)
5: asignar aptitud(R_t)
6: $\bar{P}_{t+1}$ = selección entorno(R_t)
7: P_{t+1} = reproducir ($\bar{P}_{t+1}, P_t$)
8: **Si** $local \neq \emptyset$ **entonces**
9: **Si** aleatorio entre $(0,1) < r$ **entonces**
10: búsqueda local($\bar{P}_{t+1}$)
11: **Fin si**
12: **Fin si**
13: Incrementar t en 1
14: **Fin mientras**
15: **Retornar** $\bar{P}_{t+1}$

Figura 12: Pseudocódigo algoritmo SPEA-2.

Para tener una descripción en detalle de los métodos de asignación de aptitud y reproducción se recomienda ver [24].

Pseudocódigo Método: Selección Entorno

Entrada: Población R; Tamaño población externa $\bar{N}$
Salida: Población seleccionada $sobrevivientes$

1: $sobrevivientes = \emptyset$
2: **Para** $i = 0$ **hasta** $|R|$ **incremento** en 1 **hacer**
3: **Si** R_i.aptitud < 1.0 **entonces**
4: $sobrevivientes = sobrevivientes \cup R_i$
5: Eliminar R_i
6: **Fin si**
7: **Fin para**
8: **Si** $|sobrevivientes| < \bar{N}$ **entonces**
9: ordenar (R)
10: **Mientras** $|sobrevivientes| < \bar{N}$ hacer
11: $sobrevivientes = sobrevivientes \cup R_0$

12: Eliminar R_0
13: **Fin mientras**
14: **Si no Si** $|sobrevivientes| > \bar{N}$ **entonces**
15: **Mientras** $|sobrevivientes| > \bar{N}$ hacer
16: $\boldsymbol{i}$ = índice solución más apiñada($sobrevivientes$)
17: Eliminar $sobrevivientes_i$
18: **Fin mientras**
19: **Fin si**
20: **Fin si**
21: **Retornar** $\boldsymbol{sobrevivientes}$

Figura 13: Pseudocódigo Método Selección Entorno.

4.12 PROPUESTA MEMÉTICA DE LOS ALGORITMOS MULTI-OBJETIVO

En el presente apartado se describe la forma cómo se convirtieron los algoritmos multi-objetivo (anteriormente definidos) a un enfoque memético haciendo uso de los algoritmos de búsqueda local, también expuestos previamente. El proceso de exploración del espacio de búsqueda será realizado por los algoritmos multi-objetivo (NSGA-II, MOGBHS o SPEA-2), mientras que alguno de los algoritmos de búsqueda local (HC, SA o ILS) será el encargado de realizar una búsqueda intensiva en las regiones prometedoras que pueden contener soluciones óptimas escogidas previamente por el algoritmo de exploración. El conocimiento del problema fue aplicado al proceso de generación de los nuevos individuos, alterando un porcentaje de genes dependiendo del modelo a ser calibrado, esto debido a que CORSIM no detecta cambios pequeños en los parámetros de un individuo, pero al alterar muchos parámetros de un individuo, se toma el riesgo de realizar una búsqueda al azar y no en el vecindario de la solución actual.

La Figura 14 detalla el pseudocódigo del método utilizado para realizar la explotación.

4.12.1 PROPUESTA MEMÉTICA DEL ALGORITMO NSGA-II

La explotación en NSGA-II es realizada una vez se ha terminado el ordenamiento de no dominancia, es decir, cuando el número del ranking es asignado a cada individuo de la población actual, debido a que la propuesta planteada realiza la explotación únicamente a una cantidad determinada del frente de Pareto (ranking = 0). Dado el hecho que la explotación de todo el frente aumenta drásticamente el tiempo de ejecución y además todos los individuos tendrían a converger a los mismos valores de volumen y velocidad. Una probabilidad es definida para determinar si en la iteración actual del algoritmo se realiza o no explotación.

4.12.2 PROPUESTA MEMÉTICA DEL ALGORITMO MOGBHS

La explotación se realiza después de haber evaluado y ordenado la nueva memoria armónica, es decir, después de haber agregado la nueva armonía, eliminado la peor armonía de la memoria y recalculado el frente de Pareto. En cada iteración, se determina (por medio de una probabilidad previamente definida) si se realiza o no explotación de una cantidad definida de armonías (individuos) del frente de Pareto.

4.12.3 PROPUESTA MEMÉTICA DEL ALGORITMO SPEA2

La explotación se realiza después de haber generado la siguiente población externa ($\bar{P}$), debido a que esta contiene a los mejores individuos de la población y por lo tanto los valores de aptitud ya han sido calculados. En cada iteración, se determina (por medio de una probabilidad previamente definida) si se realiza o no explotación de una cantidad definida de individuos del frente de Pareto.

4.12.4 MÉTODO DE EXPLOTACIÓN

El método que se utilizó en el presente trabajo, consiste en elegir de manera aleatoria un porcentaje de los individuos del frente de Pareto, posterior a esto, ejecutar el algoritmo de búsqueda local o algoritmo de explotación sobre los individuos seleccionados y así poder mejorar algunas soluciones del frente de Pareto. A continuación, se presenta el pseudocódigo del método de explotación o búsqueda local empleado.

Pseudocódigo Método: BusquedaLocal

Entrada: Frente de Pareto F, Porcentaje de selección δ
Salida: seleccionados

```
1: seleccionados = ∅
2: Para i = 0 hasta |F| incremento en 1 hacer
3:     Si Aleatorio entre (0, 1) < δ entonces
4:         seleccionados = seleccionados ∪ F_i
5:     Fin si
6: Fin para
7: EjecutarBusquedaLocal (seleccionados)
8: Retornar seleccionados
```

Figura 14: Pseudocódigo Método BusquedaLocal.

4.13 USO DE HILOS

En este trabajo de investigación se usaron hilos para mejorar algunos aspectos de los algoritmos elegidos los cuales son explicados delante de este párrafo, así como para el cálculo de los valores de aptitud de los individuos y para la ejecución paralela de los algoritmos de explotación. Cabe aclarar que el uso de hilos conlleva un aumento en la memoria RAM (para disminuir los problemas de inconsistencia de

memoria compartida) y un aumento en el gasto temporal de la memoria no volátil del sistema, por lo tanto, cada hilo en ejecución, tiene ligado un espacio en memoria RAM y una copia del archivo ".trf". Con el fin de realizar una simulación sin afectar a los demás hilos, el control de la cantidad máxima de ejecución de hilos en paralelo, es definida por el usuario (quien será responsable de saber cuántos hilos puede soportar su equipo).

4.13.1 HILOS EN LA EVALUACIÓN DE LA APTITUD

El cálculo de los valores de aptitud de un individuo supone el reemplazo de los parámetros del individuo sobre un archivo ".trf" y su posterior simulación en CORSIM, este proceso lleva a cabo un tiempo de ejecución proporcional al tamaño del modelo (es decir, entre más grande sea el modelo a simular, mayor será el tiempo de ejecución del proceso del cálculo de la aptitud de un individuo). Esto hace evidente el hecho de que entre más individuos se tengan en una población, mayor es el tiempo de cálculo de los distintos valores de aptitud para cada individuo. En respuesta a este inconveniente, se realizó la implementación de hilos en el proceso de evaluar una población o conjunto de individuos, logrando disminuir el tiempo que requiere ejecutar todas las simulaciones de los distintos modelos generados por una nueva población. Además de guardar la información de las aptitudes de los individuos con el fin de evitar realizar simulaciones sobre individuos previamente evaluados.

4.13.2 HILOS EN LOS ALGORITMOS PROPUESTOS

Los algoritmos multi-objetivo utilizados en este trabajo, son poblacionales, por lo tanto, se utilizan hilos en la evaluación de la aptitud de las poblaciones generadas en cada iteración, de la forma descrita anteriormente.

En el caso del algoritmo MOGBHS, se presenta una situación particular, en cada iteración del algoritmo, solamente se crea un nuevo individuo (armonía) lo cual crea una dependencia entre la velocidad de un procesador y el tiempo de ejecución de una iteración, esto genera un desuso de los demás procesadores del equipo y esto no permite su comparación en igualdad de condiciones; para solucionar este problema, se realizó una generación paralela de n individuos, logrando aprovechar los demás procesadores del equipo y además obtener mejores soluciones debido al aumento en la exploración del espacio de búsqueda en cada iteración. El valor de n corresponde a la cantidad máxima de hilos definida por el usuario.

En los algoritmos meméticos multi-objetivo, se utilizó hilos tanto en los procesos descritos anteriormente, como en el proceso de realizar la explotación a los individuos seleccionados del frente de Pareto. En el proceso de explotación, cada hilo ejecuta una copia en memoria RAM del algoritmo de búsqueda local (previamente elegido), junto con una copia temporal del archivo ".trf" para ser simulado con CORSIM y un individuo perteneciente al conjunto de seleccionados. La cantidad de ejecución de hilos en paralelo depende de la cantidad máxima de hilos definidas por el usuario (n), además de la cantidad de individuos seleccionados (θ, cuyo valor máximo es n), si esta cantidad es igual a n, se ejecutan θ copias del

algoritmo de búsqueda local elegido, y cada copia generará una solución vecina en cada iteración; finalmente, si θ es menor que n, se ejecutan $\theta - 1$ copias del algoritmo de búsqueda local y cada copia genera (de forma paralela), una cantidad de soluciones vecinas que equivale al cociente de la división entera n/θ, la copia restante genera una cantidad de soluciones vecinas equivalentes a la suma del cociente y el residuo de la anterior división.

Con el objetivo de obtener una comparación justa se implementa una estrategia de uso de hilos que consiste en mantener un equilibrio conservando una cantidad fija de hilos en los procesos de ejecución de cada algoritmo, para ello, los algoritmos de búsqueda local implementan una estrategia similar a la usada en el algoritmo MOGBHS, generando una cantidad n de soluciones vecinas de forma paralela en cada iteración. De igual manera el algoritmo GASA ejecuta el algoritmo de explotación SA lanzando en cada iteración n soluciones vecinas. Por otro lado el algoritmo SPSA (algoritmo del estado del arte con quien se realiza la comparación) genera de forma paralela $n/3$ soluciones vecinas y cada una de ellas ejecuta, también de forma paralela, tres (3) evaluaciones correspondientes a los modelos que genera el algoritmo para su funcionamiento, logrando así, al igual que los anteriores, una cantidad n de hilos simultáneos en cada proceso.

CAPÍTULO 4

5. EXPERIMENTACIÓN

En este capítulo se presentan los modelos utilizados para la evaluación y comparación de los algoritmos propuestos, para tal fin se presentan los algoritmos y medidas con los cuales se realiza la comparación. Para lograr un procedimiento eficiente (tiempo de ejecución) y obtener mejores resultados se presenta el proceso utilizado para realizar un afinamiento de parámetros de los dos algoritmos que obtuvieron mejor comportamiento en evaluaciones previas, Luego de esto se muestran los resultados obtenidos, junto a un minucioso análisis del mismo. Por último, se expone el análisis de sensibilidad del algoritmo ganador.

La ejecución de los experimentos fue realizada sobre un servidor provisto por la Universidad de Nevada, con sede en Las Vegas, Estados Unidos, el cual consta de las siguientes características: Sistema operativo: Windows Server 2008 R2 de 64 bits; Procesador: Intel Xeon CPU X7560 2.26 GHz de 64 núcleos; Memoria RAM: 256 Gigabytes DDR3; Software: Java Virtual Machine V1.7.60.

5.1 MODELOS DE PRUEBA

Tres (3) modelos de diferentes tamaños y complejidades computacionales fueron utilizados para la evaluación de los algoritmos, la complejidad de cada modelo fue definida de la misma manera que en [6], clasificando cada uno con un nivel de complejidad (baja, media, alta) dependiendo de la cantidad de enlaces que tenga cada modelo, así, un modelo es de complejidad baja si tiene entre 0 y 99 enlaces, media entre 100 y 300 enlaces; y alta si cuenta con más de 300 enlaces.

Como se mencionó anteriormente los parámetros de los diferentes modelos que se van a calibrar son el comportamiento del conductor y el rendimiento del vehículo, los datos de salida que se obtienen a partir de la simulación de los modelos son los valores del conteo de vehículos (volumen) medidos en la cantidad de vehículos por hora (vph) y la velocidad de los mismos medida en millas por hora (mph). Entre menor sea la diferencia entre estos valores simulados y los obtenidos en campo (reales), se considera mejor el conjunto de parámetros calibrados. Los parámetros a calibrar son los mencionados en [1] los cuales fueron utilizados en [6].

Los modelos presentados a continuación y los datos de campo para los valores de volumen y velocidad utilizados fueron suministrados por el Departamento de transporte de Nevada NDOT (Nevada Department of Transportation).

5.1.1 MODELO DE COMPLEJIDAD BAJA: MCTRANS NETWORK

Esta red es una red artificial proporcionada por McTrans Center, los parámetros por defecto son tomados para la simulación del modelo. Los resultados de esta simulación se asignan como los datos de campo para el experimento. Todos los

parámetros de calibración se modificaron al azar. Este modelo modificado se usó como punto de partida para la calibración. Dicho modelo incluye un total de 20 enlaces arteriales; los recuentos y la velocidad del vehículo se utilizan simultáneamente para realizar la calibración. El valor inicial de NRMS (ecuación (3)) en esta red es de 0,29118. La Figura 15 ilustra el modelo CORSIM a calibrar.

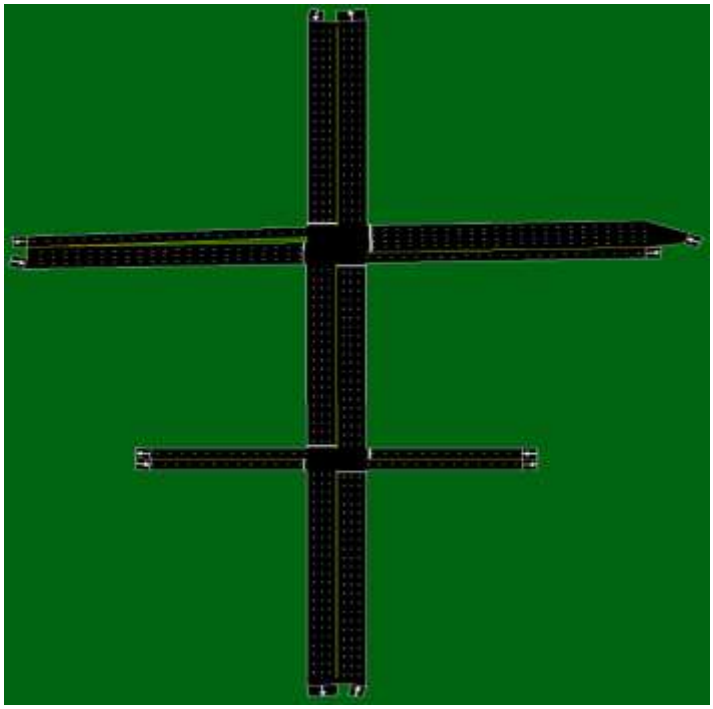

Figura 15: Imagen generada con el software TRAFVU Viewer a partir del modelo CORSIM de la red McTrans.

5.1.2 MODELO DE COMPLEJIDAD MEDIA: RENO NETWORK

El experimento de complejidad media contiene un modelo CORSIM de la autopista Pyramid en Reno, Nevada, Estados Unidos. Los datos para el conteo de vehículos (volumen) y velocidad fueron medidos en campo tomados por sensores ubicados en la autopista. Este modelo incluye un total de 126 enlaces arteriales de los cuales se dispone de datos para 45 de estos. El valor inicial de NRMS en esta red es de 0,22179. La Figura 16 muestra en la parte izquierda el mapa de Google y en la derecha el modelo CORSIM de la autopista Pyramid.

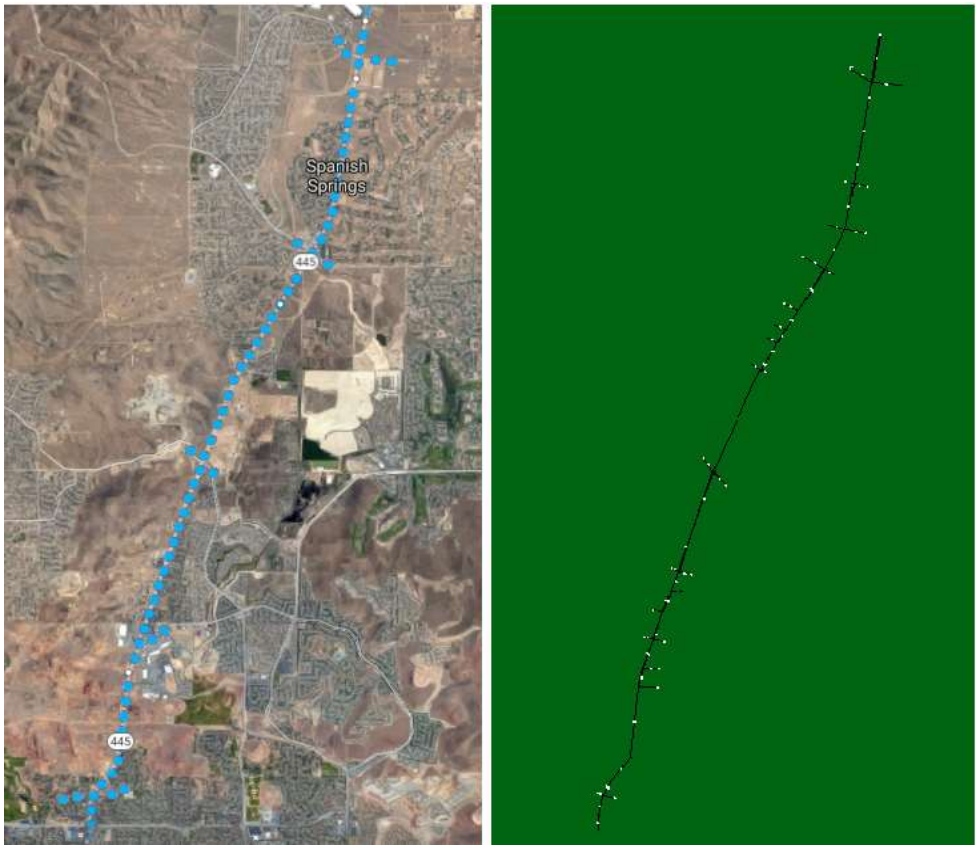

Figura 16: Autopista Pyramid, Reno, Nevada. Tomado de Google Maps y TRAFVU Viewer.

5.1.3 MODELO DE COMPLEJIDAD ALTA: I-75 NETWORK

Este experimento contiene un modelo CORSIM que representa una porción de la autopista interestatal I-75 en Miami, Florida, Estados Unidos. Al igual que el modelo de complejidad media, los datos para el conteo de vehículos fueron medidos en campo a través de sensores mientras que los datos de campo para velocidad fueron simulados a partir de los parámetros por defecto debido a que no se cuenta con valores de campo para la velocidad en esta red. Este modelo incluye un total de 703 enlaces de los cuales se poseen datos de campo para 346 de ellos. El valor inicial de NRMS en esta red es de 0,33182. La Figura 17 muestra en la parte izquierda el mapa de Google y en la derecha el modelo CORSIM de la autopista I-75.

Figura 17: Autopista I-75, Miami, FL. Tomado de Google Maps y TRAFVU Viewer.

5.2 ALGORITMOS A COMPARAR Y MEDIDAS

Con el propósito de mostrar las ventajas de los algoritmos propuestos en el problema de calibración de modelos de flujo de tráfico vehicular CORSIM, se presenta una comparación frente a dos algoritmos del estado del arte, SPSA y GASA que han mostrado los mejores resultados en este tipo de problemas, además se realiza una comparación con el algoritmo SPEA-2 que presenta buenos resultados en diferentes problemas de optimización multi-objetivo. La comparación se realiza con los promedios de tiempo de ejecución y la medida NRMS.

5.3 AFINAMIENTO DE PARÁMETROS

Con el propósito de obtener un procedimiento eficiente (tiempo de ejecución) y lograr mejores resultados se realiza un afinamiento sobre los parámetros de los dos algoritmos que obtuvieron mejor comportamiento en evaluaciones previas. Estos dos algoritmos fueron obtenidos a partir de la evaluación de todos los algoritmos propuestos (en sus versiones multi-objetivo y memético multi-objetivo) ejecutándolos 30 veces sobre los modelos con los parámetros recomendados en la literatura para cada algoritmo. Una vez obtenidos los resultados de cada algoritmo, se realizaron sobre estos, análisis estadísticos basados en las pruebas de Wilcoxon y Friedman utilizando la herramienta Keel Software [84]. Este análisis arrojó que los

algoritmos con mejor comportamiento fueron: MOGBHS y la propuesta memética del NSGA-II utilizando el algoritmo de búsqueda local ILS.

El afinamiento de los parámetros de los algoritmos se hizo mediante arreglos de cobertura (CA), se optó por escoger un CA debido a que estos se han utilizado exitosamente para automatizar la generación de pruebas de software y en el proceso de afinamiento permiten disminuir el número de pruebas. Se utilizó un CA de fuerza 2 y tamaño de vocabulario 5, los valores para el vocabulario de cada algoritmo fueron obtenidos mediante una ligera variación de los parámetros recomendados en la literatura. Por último, la cantidad de parámetros no es la misma en cada algoritmo por tanto se hizo necesario realizar un CA para cada algoritmo. La Tabla 1 y la Tabla 2 ilustran los valores del vocabulario para los algoritmos MOGBHS y NSGA-II con ILS respectivamente. En la primera fila se encuentran los valores con los que se realizó la evaluación previa, es decir los valores recomendados en la literatura. Los CA utilizados se muestran a continuación de estas tablas.

PARmin	PARmax	HMS	HMCR	Improvisaciones
0,1	0,9	40	0,95	450
0,05	0,95	10	0,97	550
0	1	20	0,9	500
0,15	0,85	60	0,85	400
0,2	0,8	80	0,8	360

Tabla 1: Vocabulario para MOGBHS.

Iteraciones	Tamaño población	Probabilidad Aceptación	Probabilidad Mutación	Radio	Aceptación	Porcentaje Explotación	Probabilidad Explotación	Iteraciones LS	Porcentaje Mutación
50	40	0,4	0,1	0,9	0,35	0,5	0,25	30	0,1
70	10	0,5	0,25	0,95	0,5	0,7	0,5	50	0,15
60	20	0,3	0,2	0,8	0,4	0,6	0,4	45	0,07
40	60	0,2	0,15	0,7	0,3	0,4	0,3	40	0,05
30	80	0,1	0,05	0,6	0,25	0,3	0,2	20	0,03

Tabla 2: Vocabulario para NSGA-II con ILS.

La Tabla 3 y la Tabla 4 indican el CA resultante para los algoritmos MOGBHS y NSGA-II con ILS respectivamente, el número de filas indican la cantidad de casos de prueba que se deben realizar para cada algoritmo (33 para MOGBHS y 45 para NSGA-II con ILS), cada caso de prueba fue ejecutado 30 veces sobre el modelo de complejidad baja con el objetivo de seleccionar la mejor configuración.

PARmin	PARmax	HMS	HMCR	Improvisaciones
0,1	0,9	20	0,97	400
0,1	0,95	40	0,9	360
0,1	1	10	0,8	550
0,1	0,85	60	0,95	360

0,1	0,8	60	0,85	450
0,05	0,9	80	0,85	360
0,05	0,95	10	0,97	400
0,05	1	20	0,9	500
0,05	0,85	40	0,97	450
0,05	0,8	60	0,8	400
0	0,9	10	0,95	450
0	0,95	20	0,95	550
0	1	60	0,97	360
0	0,85	80	0,9	400
0	0,8	40	0,85	360
0,15	0,9	40	0,8	400
0,15	0,95	60	0,85	500
0,15	1	80	0,95	450
0,15	0,85	20	0,85	550
0,15	0,8	10	0,9	360
0,2	0,9	60	0,9	550
0,2	0,95	80	0,8	450
0,2	1	40	0,95	400
0,2	0,85	10	0,85	500
0,2	0,8	20	0,97	360
0,1	0,8	80	0,97	500
0,05	0,8	80	0,95	550
0,15	0,9	40	0,97	550
0	0,85	20	0,8	500
0,1	1	40	0,85	400
0,1	0,9	40	0,95	500
0,1	0,9	20	0,9	450
0,1	0,9	40	0,8	360

Tabla 3: CA para MOGBHS.

Iteraciones	Tamaño población	Probabilidad Aceptación	Probabilidad Mutación	Radio	Aceptación	Porcentaje Explotación	Probabilidad Explotación	Iteraciones LS	Porcentaje Mutación
50	40	0,3	0,25	0,7	0,35	0,6	0,2	20	0,05
50	10	0,4	0,2	0,6	0,5	0,4	0,2	20	0,15
50	20	0,5	0,05	0,95	0,3	0,3	0,25	45	0,1
50	60	0,2	0,1	0,6	0,35	0,5	0,25	50	0,1
50	80	0,2	0,15	0,9	0,3	0,7	0,2	40	0,07
70	40	0,1	0,15	0,6	0,3	0,7	0,5	45	0,07
70	10	0,5	0,25	0,7	0,4	0,5	0,3	50	0,07
70	20	0,3	0,2	0,8	0,25	0,7	0,3	45	0,05
70	60	0,4	0,25	0,9	0,3	0,3	0,5	30	0,03

70	80	0,2	0,05	0,7	0,5	0,4	0,4	20	0,03
60	40	0,5	0,1	0,9	0,25	0,7	0,4	50	0,15
60	10	0,3	0,1	0,95	0,5	0,3	0,5	40	0,05
60	20	0,2	0,25	0,6	0,5	0,5	0,4	45	0,03
60	60	0,1	0,2	0,7	0,3	0,3	0,25	50	0,1
60	80	0,4	0,15	0,6	0,4	0,6	0,3	45	0,1
40	40	0,4	0,05	0,7	0,25	0,5	0,5	40	0,1
40	10	0,2	0,15	0,8	0,25	0,3	0,2	45	0,15
40	20	0,1	0,1	0,9	0,4	0,6	0,5	20	0,15
40	60	0,3	0,15	0,95	0,3	0,4	0,4	50	0,05
40	80	0,5	0,2	0,6	0,35	0,4	0,5	40	0,03
30	40	0,2	0,2	0,95	0,4	0,6	0,4	40	0,03
30	10	0,1	0,05	0,9	0,35	0,7	0,3	45	0,03
30	20	0,4	0,1	0,7	0,35	0,3	0,4	45	0,07
30	60	0,5	0,15	0,8	0,5	0,6	0,5	50	0,03
30	80	0,3	0,25	0,6	0,25	0,3	0,2	20	0,1
50	80	0,1	0,25	0,8	0,25	0,4	0,25	40	0,03
70	80	0,1	0,1	0,95	0,3	0,5	0,2	40	0,15
40	80	0,4	0,25	0,95	0,35	0,7	0,25	30	0,15
60	60	0,3	0,05	0,8	0,4	0,4	0,2	50	0,07
70	20	0,5	0,15	0,7	0,35	0,4	0,2	30	0,1
30	40	0,4	0,1	0,8	0,3	0,4	0,3	30	0,03
40	40	0,3	0,2	0,9	0,5	0,5	0,25	30	0,07
60	10	0,1	0,05	0,6	0,35	0,6	0,4	30	0,05
50	40	0,2	0,1	0,9	0,4	0,3	0,3	30	0,05
70	10	0,5	0,1	0,9	0,3	0,6	0,25	20	0,1
40	60	0,4	0,1	0,95	0,25	0,7	0,3	20	0,07
50	60	0,1	0,05	0,7	0,5	0,7	0,3	40	0,15
30	40	0,5	0,15	0,8	0,35	0,5	0,25	20	0,05
50	60	0,2	0,1	0,9	0,5	0,4	0,5	45	0,1
60	40	0,4	0,1	0,9	0,4	0,7	0,25	20	0,1
50	40	0,4	0,1	0,8	0,25	0,6	0,4	30	0,1
50	80	0,4	0,1	0,9	0,35	0,5	0,25	50	0,05
50	20	0,3	0,1	0,9	0,35	0,5	0,25	50	0,15
50	20	0,4	0,1	0,9	0,35	0,5	0,25	40	0,03
50	40	0,4	0,1	0,9	0,35	0,6	0,25	30	0,07

Tabla 4: CA para NSGA-II con ILS.

La decisión de realizar un afinamiento de parámetros solo a dos algoritmos, de no hacer un diseño experimental completo ($t = k$ donde t es la fuerza y k la cantidad de parámetros del algoritmo) y de no hacer las pruebas sobre todos los modelos, fue tomada debido a que cada ejecución del algoritmo conlleva un tiempo de

ejecución muy alto y eso a su vez implica extender los tiempos del proyecto más allá de lo definido para un trabajo de grado de la FIET en la Universidad del Cauca. Finalmente, los resultados arrojados por los CA otorgan los mejores parámetros para el conjunto de pruebas realizadas para estos algoritmos y ofrecen una base más sólida para los parámetros de los demás algoritmos, en comparación con los parámetros utilizados anteriormente.

5.4 RESULTADOS OBTENIDOS Y COMPARACIÓN

A continuación se muestran los resultados obtenidos por los algoritmos sobre los tres modelos de prueba; cada algoritmo fue ejecutado 30 veces en cada modelo con el propósito de tener un comportamiento promedio sobre los resultados. Para la comparación con los algoritmos mono-objetivo se utilizó un peso W de 0.7 (70%) para los datos de volumen y 0.3 (30%) para velocidad, estos valores son los mismos utilizados en [6], lo cuales son provistos por expertos en tráfico de NDOT. Esto valores se deben, entre otros, a que los datos de velocidad afectan a pocos de los parámetros de los modelos utilizados.

Con el propósito de tener un equilibrio en la cantidad de evaluaciones de la función objetivo (EFO) los algoritmos de búsqueda local implementan una estrategia en donde en cada iteración en lugar de generar un individuo vecino, generan una cantidad (n) que depende de la cantidad de hilos con la que se desee trabajar. Los valores para los parámetros del algoritmo búsqueda local iterada (ILS) fueron obtenidos a partir de la ejecución y análisis del arreglo de cobertura del algoritmo memético NSGA-II-ILS, igualmente para el algoritmo ascenso a la colina (HC), dado que el algoritmo ILS utiliza como algoritmo local el algoritmo HC. Finalmente los valores para los parámetros del algoritmo de búsqueda local recocido simulado SA fueron los utilizados en el estado del arte. Los valores de los parámetros para la ejecución de los algoritmos de búsqueda local son los siguientes:

- Ascenso a la colina: Radio = 0,9; Porcentaje perturbar = 10%.
- Recocido simulado: Temperatura = 15; Taza de enfriamiento = 0,95; Porcentaje cambio = 30%; Constante aceptación = 0,35.
- Búsqueda local iterada: Porcentaje perturbar = 10%; Probabilidad aceptación = 0.4; Constante de aceptación = 0,35.

El número de iteraciones para cada uno de los algoritmos de búsqueda local es de 364. Cuando se incluyen los algoritmos de búsqueda local a los multi-objetivos, la cantidad de iteraciones de estos es de 20, cada vez que se realiza explotación. La cantidad de hilos para todos los algoritmos propuestos (búsqueda local, multi-objetivo y memético multi-objetivo) es fijada en quince (15).

5.4.1 RESULTADOS Y COMPARACIÓN PARA EL MODELO MCTRANS

5.4.1.1 RESULTADOS OBTENIDOS CON NSGA-II

A continuación se presentan los resultados obtenidos en el modelo de complejidad baja por el algoritmo multi-objetivo NSGA-II y sus propuestas meméticas. Los parámetros utilizados para la ejecución del algoritmo NSGA-II fueron obtenidos a partir de la ejecución y análisis del arreglo de cobertura del algoritmo memético NSGA-II-ILS, realizando una pequeña modificación en el número de iteraciones y tamaño de la población, con el objetivo de que todos los algoritmos realicen el mismo número de evaluaciones de la función objetivo y tener una similitud en la cantidad de hilos en ejecución. Los parámetros fueron los siguientes: Tamaño de la población = 60; Número de generaciones (iteraciones) = 180; Porcentaje de mutación = 10%. Para la propuesta memética se utiliza la combinación de los parámetros de los algoritmos NSGA-II y de búsqueda local descritos anteriormente con la única diferencia de que la cantidad de generaciones es de 60, esto dado que las iteraciones internas del algoritmo de explotación incrementan el tiempo de ejecución y la cantidad de EFO. Por otro lado también se fija una probabilidad de realizar explotación del 20% y un porcentaje de explotación del 50%.

La Figura 18 muestra el tiempo de ejecución (eje X) y el resultado obtenido del NRMS (eje Y), del promedio de las 30 ejecuciones de los algoritmos: HC, NSGA-II y la propuesta memética del NSGA-II con HC denominado NSGA-II-HC. Se logra observar como el algoritmo HC tiende a converger rápidamente, pero este se estanca en un óptimo local, lo cual se puede ver a lo largo del tiempo, el valor promedio de NRMS para este algoritmo es igual a 0,0257 lo que equivale a una mejora de aproximadamente 91%. Por otro lado se puede apreciar que el algoritmo multi-objetivo NSGA-II tiene un comportamiento similar al HC con la diferencia que NSGA-II no tiende a converge tan rápidamente, esto dada su naturaleza de amplia exploración en el espacio de búsqueda, este algoritmo obtiene un valor promedio de NRMS igual a 0,0232 correspondiendo a una mejora del 92% aproximadamente. El tiempo de ejecución de este algoritmo fue el mayor de los 3. Finalmente la propuesta memética NSGA-II-HC es quien logra mejores resultados pero su comportamiento promedio es desequilibrado (desequilibrado refiriéndose a la alta dispersión de los puntos y equilibrado refiriéndose a la poca dispersión de los puntos) como los anteriores algoritmos, el tiempo de ejecución de este algoritmo es muy parecido al algoritmo HC. El valor promedio de NRMS para la propuesta memética es de 0,0219 que equivale a una mejora de aproximadamente 92,5%.

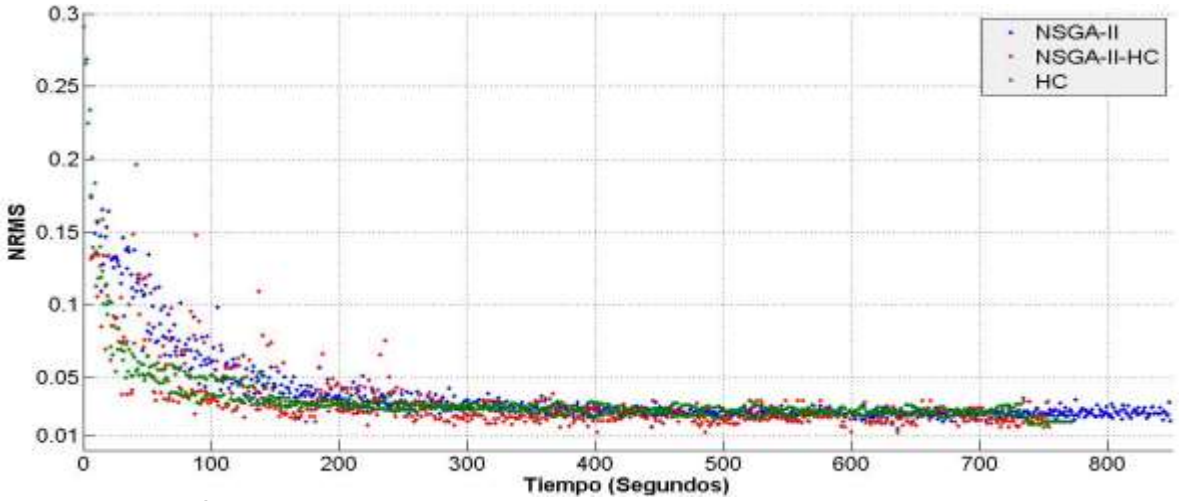

Figura 18: Gráfica NRMS vs Tiempo en la red McTrans del algoritmo NSGA-II y su propuesta memética que incluye HC.

La Figura 19 muestra el promedio de las 30 ejecuciones de los algoritmos: SA, NSGA-II y la propuesta memética del NSGA-II con SA denominado NSGA-II-SA. Al igual que el HC, el algoritmo SA tiende a converger rápidamente pero se estanca en un óptimo local, su valor de NRMS promedio es de 0,0239 lo que equivale a una mejora de aproximadamente 91,8%. Los comportamientos del algoritmo multi-objetivo y memético multi-objetivo son muy similares al de búsqueda local. Se puede apreciar, además, como el algoritmo NSGA-II-SA se comporta de manera más equilibrada en comparación con la propuesta memética NSGA-II-HC, esto se debe a la estrategia del SA de salir de óptimos locales. El valor de NRMS promedio para la propuesta memética es de 0,0223 lo cual corresponde a una mejora de aproximadamente el 92,3%. Finalmente el tiempo de ejecución de la propuesta memética fue el menor de los 3.

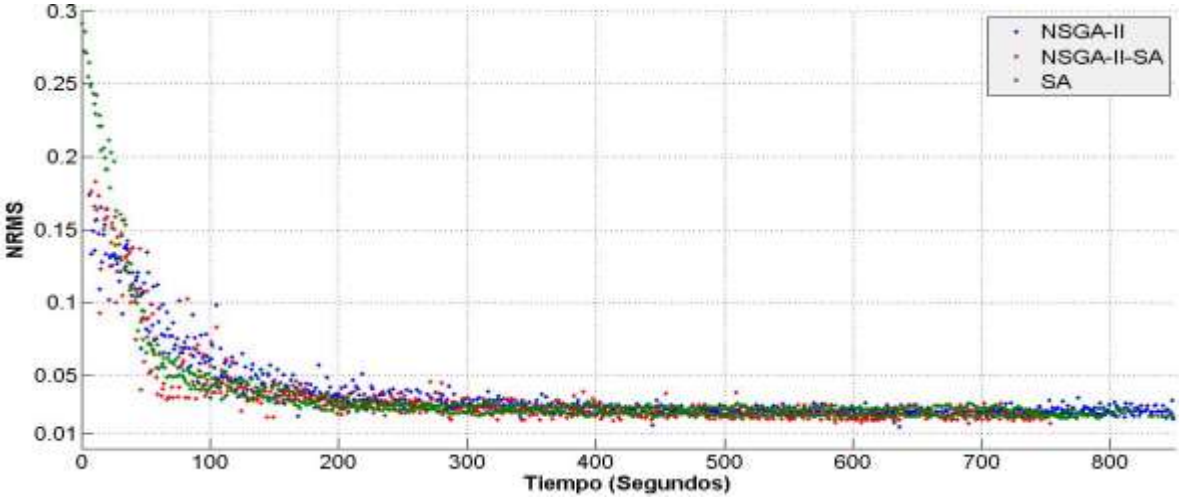

Figura 19: Gráfica NRMS vs Tiempo en la red McTrans del algoritmo NSGA-II y su propuesta memética que incluye SA.

La Figura 20 muestra el promedio de las 30 ejecuciones de los algoritmos: ILS, NSGA-II y la propuesta memética del NSGA-II con ILS denominado NSGA-II-ILS. Al

igual que los anteriores algoritmos de búsqueda local, el ILS tiende a converger rápidamente con la diferencia que este no se estanca en un óptimo local, el valor de NRMS promedio para este algoritmo de búsqueda local es de 0,0239 lo que equivale a una mejora de aproximadamente 91,8%, además este algoritmo tiene un tiempo de ejecución similar a NSGA-II y NSGA-II-ILS. Correspondiente a la propuesta memética se logra apreciar como ésta, inicialmente se comporta de manera similar al algoritmo NSGA-II, obteniendo al final del tiempo una ligera mejora de NRMS llegando a un valor promedio de 0,0217 lo que equivale una mejora del 92.5% aproximadamente.

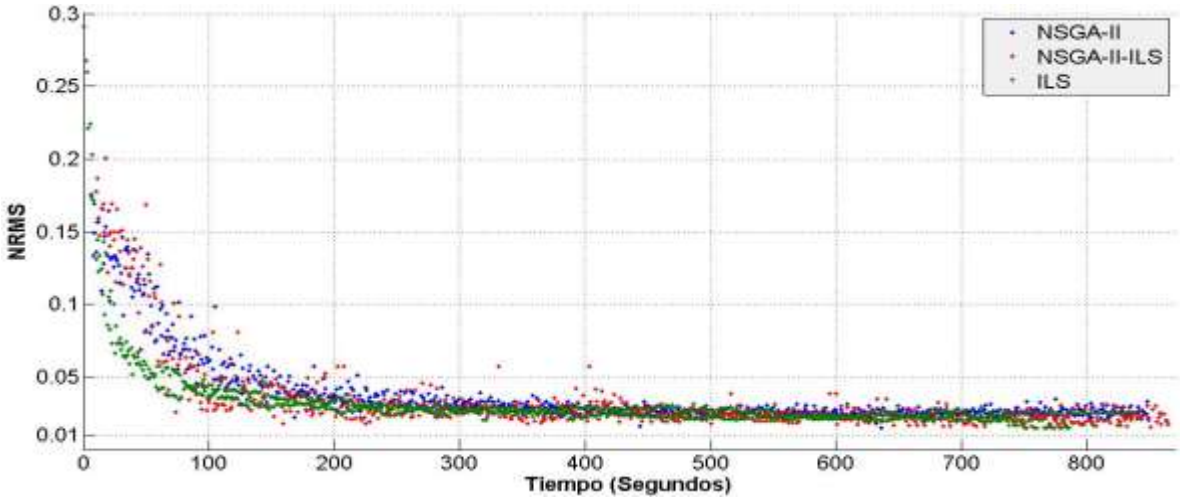

Figura 20: Gráfica NRMS vs Tiempo en la red McTrans del algoritmo NSGA-II y su propuesta memética que incluye ILS.

5.4.1.2 RESULTADOS OBTENIDOS CON MOGBHS

A continuación se presentan los resultados obtenidos en el modelo de complejidad baja por el algoritmo multi-objetivo MOGBHS y sus propuestas meméticas. Los parámetros utilizados para la ejecución del algoritmo MOGBHS fueron los obtenidos a partir de la ejecución y análisis del arreglo de cobertura. Los parámetros se presentan a continuación: PARmin = 0; PARmax = 1; HMS = 60; HMCR = 0,97; Número de improvisaciones (iteraciones) = 360. Para la propuesta memética se utiliza la combinación de los parámetros de los algoritmos MOGBHS y de búsqueda local descritos anteriormente con la única diferencia de que la cantidad de improvisaciones es de 120, esto dado que las iteraciones internas del algoritmo de explotación incrementan el tiempo de ejecución y la cantidad de EFO. Por otro lado también se fija una probabilidad de realizar explotación del 10% y un porcentaje de explotación del 50%.

La Figura 21 muestra el promedio de las 30 ejecuciones de los algoritmos: HC, MOGBHS y la propuesta memética del MOGBHS con HC denominado MOGBHS-HC. De esta figura se logra analizar que el algoritmo HC es quien obtiene mejores resultados al inicio de la ejecución, esto debido a su amplia explotación, además su tiempo de ejecución es ligeramente menor. Sin embargo, tanto el algoritmo multi-

objetivo MOGBHS como la propuesta memética MOGBHS-HC mejoran a través del tiempo demostrando que gracias a la amplia exploración del espacio de búsqueda estos algoritmos evitan estancarse en óptimos locales. De estos dos algoritmos quien presenta mejor comportamiento es el algoritmo MOGBHS dado que presenta un comportamiento más equilibrado y obtiene un mejor resultado promedio de NRMS el cual es igual a 0,0147 lo que corresponde a una mejora aproximada del 95%. Mientras que el algoritmo MOGBHS-HC obtiene un valor promedio de NRMS igual a 0,0155 correspondiendo a una mejora del 94,6% aproximadamente. Esto indica que el algoritmo de búsqueda local HC sólo contribuye un poco a la convergencia del algoritmo memético.

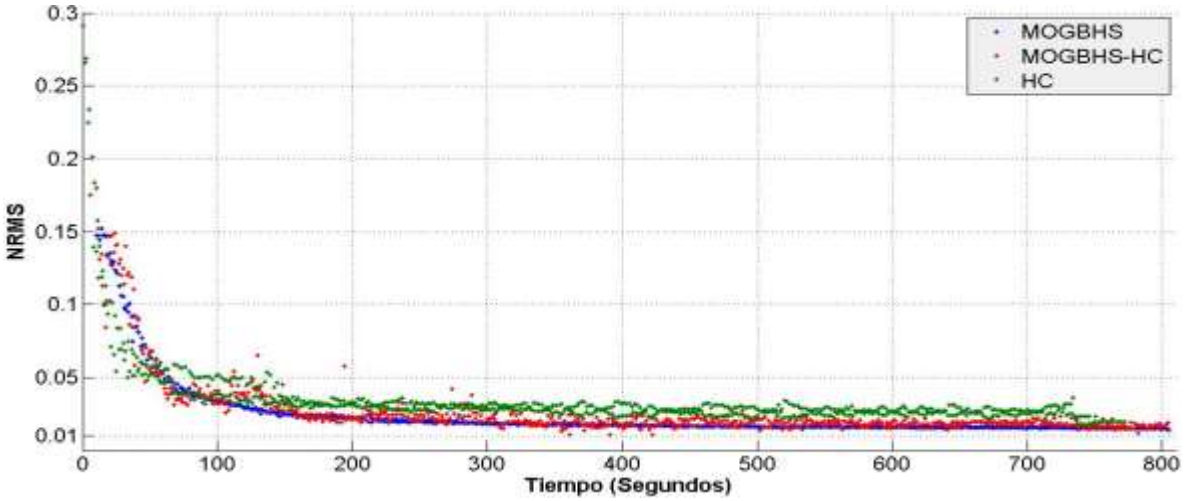

Figura 21: Gráfica NRMS vs Tiempo en la red McTrans del algoritmo MOGBHS y su propuesta memética que incluye HC.

La Figura 22 ilustra el promedio de las 30 ejecuciones de los algoritmos: SA, MOGBHS y la propuesta memética del MOGBHS con SA denominado MOGBHS-SA. Se logra observar que la propuesta memética MOGBHS-SA tiende a converger un poco más rápido, pero su comportamiento inicial no es muy equilibrado, el tiempo de ejecución de este algoritmo es bastante similar al del algoritmo MOGBHS. El valor promedio de NRMS para la propuesta memética MOGBHS-SA es de 0,0174 lo que representa una mejora del 94% aproximadamente. Esto indica que, al igual que HC, el algoritmo de búsqueda local SA sólo contribuye un poco a la convergencia del algoritmo memético.

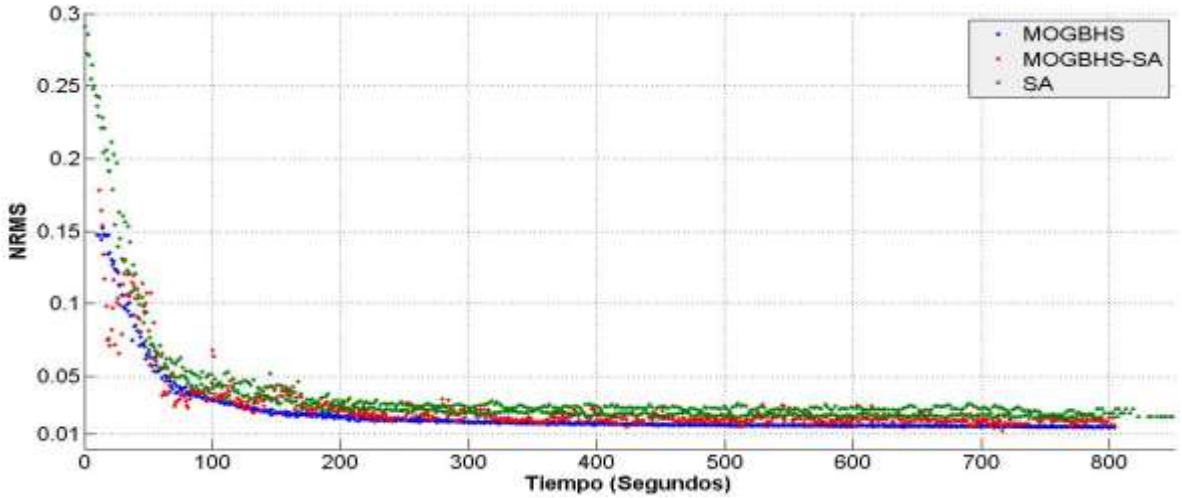

Figura 22: Gráfica NRMS vs Tiempo en la red McTrans del algoritmo MOGBHS y su propuesta memética que incluye SA.

La Figura 23 ilustra el promedio de las 30 ejecuciones de los algoritmos: ILS, MOGBHS y la propuesta memética del MOGBHS con ILS denominado MOGBHS-ILS. El algoritmo de búsqueda local ILS, al igual que el algoritmo HC, es superior que el algoritmo multi-objetivo y memético multi-objetivo al inicio de la ejecución, sin embargo su tiempo de ejecución es ligeramente mayor. Por otro lado se logra apreciar que al igual que las anteriores propuestas meméticas el MOGBHS-ILS en promedio no logra superar al algoritmo MOGBHS, lo que indica que aunque el algoritmo de búsqueda local ILS no se estanca en un óptimo local este sigue sin aportar mucho al algoritmo de búsqueda global. El valor promedio de NRMS para esta propuesta es de 0,0158 lo cual corresponde a una mejora de aproximadamente 94,6%.

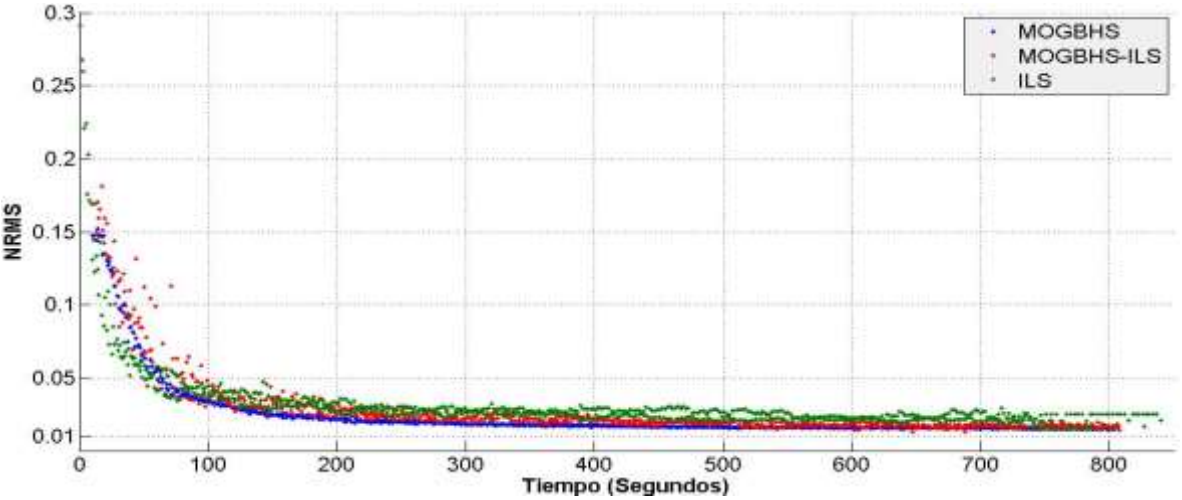

Figura 23: Gráfica NRMS vs Tiempo en la red McTrans del algoritmo MOGBHS y su propuesta memética que incluye ILS.

5.4.1.3 RESULTADOS OBTENIDOS CON SPEA-2

A continuación se presentan los resultados obtenidos en el modelo de complejidad baja por el algoritmo multi-objetivo SPEA-2 y sus propuestas meméticas. Los parámetros utilizados para la ejecución del algoritmo SPEA-2 fueron los siguientes: Tamaño de la población = 60; Tamaño de la población externa = 15; Número de generaciones (iteraciones) = 180; Porcentaje de mutación = 10%. Para la propuesta memética se utiliza la combinación de los parámetros de los algoritmos SPEA-2 y de búsqueda local descritos anteriormente con la única diferencia de que la cantidad de improvisaciones es de 60, esto debido a que las iteraciones internas del algoritmo de explotación incrementan el tiempo de ejecución y la cantidad de EFO. Por otro lado también se fija una probabilidad de realizar explotación del 20% y un porcentaje de explotación del 50%.

La Figura 24 muestra el promedio de las 30 ejecuciones de los algoritmos: HC, SPEA-2 y la propuesta memética del SPEA-2 con HC denominado SPEA-2-HC. Se puede apreciar, al igual que en NSGA-II, MOGBHS y sus propuestas meméticas con HC, que el algoritmo HC es quien obtiene mejores resultados al inicio de la ejecución, también se observa que el algoritmo HC se encuentra a la par con el algoritmo multi-objetivo SPEA-2, pero este último no converge tan rápidamente como el anterior y su tiempo de ejecución es ligeramente mayor llegando a un valor promedio de NRMS igual a 0,0247 correspondiendo a una mejora aproximada del 91,5%. Por otro lado se puede observar que el comportamiento de la propuesta memética SPEA-2-HC es ligeramente superior, su promedio de NRMS es de 0,0201 lo que equivale una mejora aproximada del 93%.

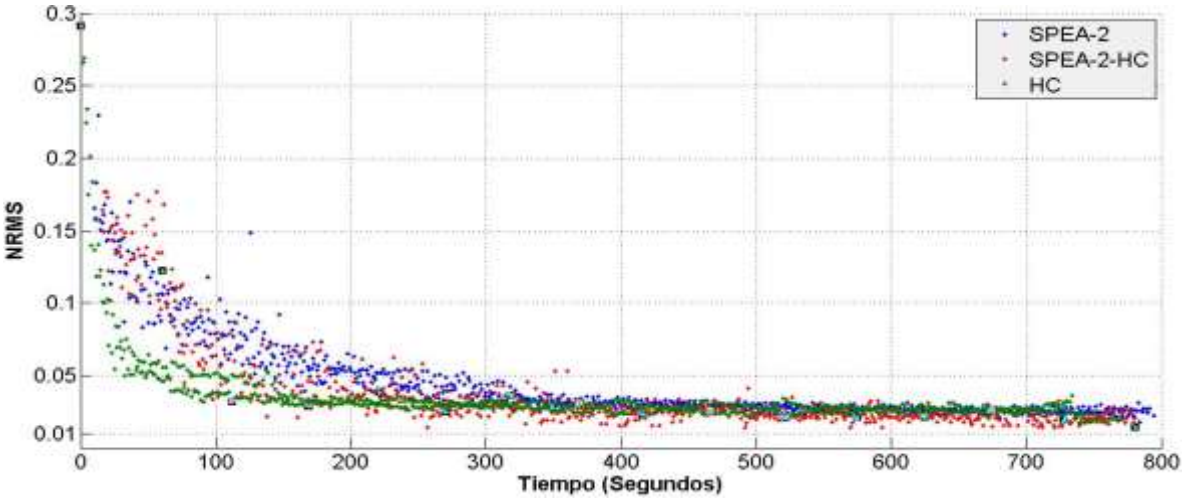

Figura 24: Gráfica NRMS vs Tiempo en la red McTrans del algoritmo SPEA-2 y su propuesta memética que incluye HC.

La Figura 25 muestra el promedio de las 30 ejecuciones de los algoritmos: SA, SPEA-2 y la propuesta memética del SPEA-2 con SA denominado SPEA-2-SA, de ella se puede apreciar que los tres algoritmos tienden a valores muy similares de NRMS y tiempo de ejecución, sin embargo el algoritmo SPEA-2 demora su convergencia, esto dado su amplia exploración en el espacio de búsqueda. La

propuesta memética SPEA-2-SA, obtiene resultados ligeramente mejores que los anteriores algoritmos y esta propuesta, a diferencia del algoritmo SPEA-2-HC tiende a converger rápidamente, incluso más rápido que el algoritmo de búsqueda local. El valor promedio de NRMS para este algoritmo es de 0,0225 correspondiendo a una mejora del 92,3%.

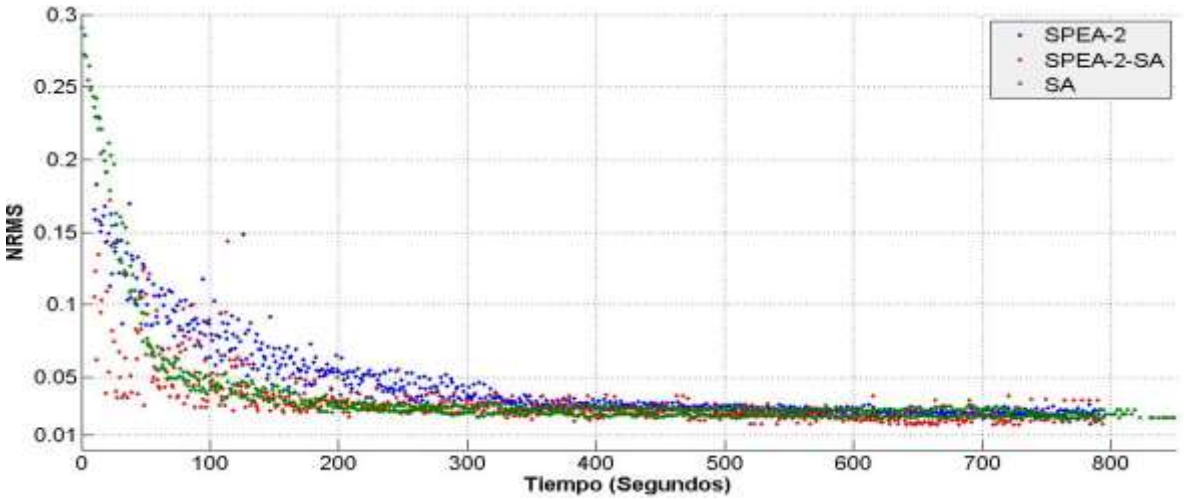

Figura 25: Gráfica NRMS vs Tiempo en la red McTrans del algoritmo SPEA-2 y su propuesta memética que incluye SA.

La Figura 26 muestra el promedio de las 30 ejecuciones de los algoritmos: ILS, SPEA-2 y la propuesta memética del SPEA-2 con ILS denominado SPEA-2-ILS, de ella se puede apreciar que, el algoritmo de búsqueda local obtiene mejor valor promedio de NRMS que el algoritmo multi-objetivo, en este caso, el SPEA-2. Con referencia a la propuesta memética SPEA-2-ILS, ésta presenta un comportamiento bastante desequilibrado al inicio de la ejecución, sin embargo a medida que la ejecución avanza su comportamiento se equilibra, e inclusive logra obtener mejores resultados de NRMS llegando a un valor promedio de 0,0187 correspondiendo una mejora del 93,5% aproximadamente.

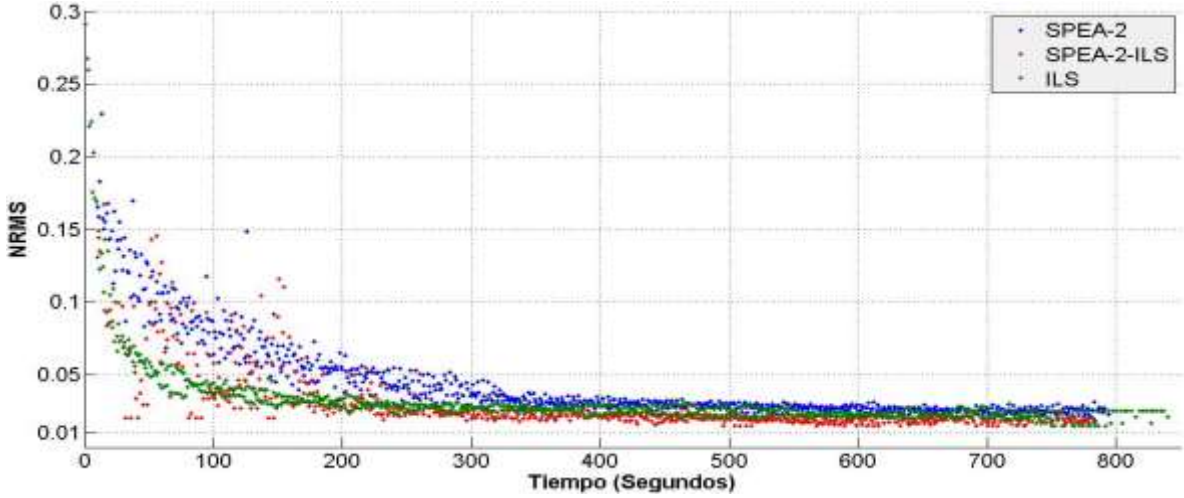

Figura 26: Gráfica NRMS vs Tiempo en la red McTrans del algoritmo SPEA-2 y su propuesta memética que incluye ILS.

La Tabla 5 muestra el resumen de los resultados obtenidos por todos los algoritmos en la red de complejidad baja. Se logra apreciar cómo el peor resultado encontrado en los algoritmos de búsqueda local es muy alto en comparación con los encontrados por los algoritmos multi-objetivo y mucho mayor en comparación con los meméticos multi-objetivo, lo que demuestra nuevamente, que estos algoritmos tienen la desventaja de estancarse en un óptimo local. De forma similar se puede apreciar como la desviación estándar es mayor en los algoritmos de búsqueda local, lo que indica que aunque el promedio de estos algoritmos es relativamente bueno, no se puede garantizar que con estos algoritmos siempre se llegue a valores cercanos al promedio. Por otra parte se destaca la inclusión de los algoritmos de búsqueda local en los algoritmos NSGA-II y SPEA-2, logrando siempre un mejor valor promedio de NRMS, mejorando, además, en la mayoría de los casos, el valor de la desviación estándar, e incluso (en el algoritmo NSGA-II) logra disminuir significativamente el tiempo de ejecución. La propuesta memética que logra mejores resultados promedio de NRMS para estos dos algoritmos multi-objetivo es la que incluye al algoritmo ILS. Finalmente, con respecto al algoritmo MOGBHS, se aprecia cómo el mejor comportamiento se logra solo con el algoritmo multi-objetivo, sin la inclusión de un algoritmo de búsqueda local, e incluso este es quien obtiene mejor valor de NRMS de todos los algoritmos expuestos en la tabla.

Algoritmo	NRMS				GEH del mejor resultado	Tiempo promedio (mm:ss)
	Mejor resultado	Peor resultado	Promedio	Desviación estándar		
HC	0,0152	0,0684	0,0257	0,01185145	100%	12:52
SA	0,0168	0,0611	0,0239	0,00765998	100%	14:09
ILS	0,0150	0,0767	0,0239	0,01123095	100%	14:00
NSGA-II	0,0137	0,0311	0,0232	0,003979053	100%	14:09
NSGA-II-HC	0,0126	0,0340	0,0219	0,005008199	100%	12:34
NSGA-II-SA	0,0134	0,0312	0,0223	0,003884468	100%	12:33
NSGA-II-ILS	**0,0126**	**0,0302**	**0,0217**	**0,004202199**	**100%**	**14:25**
MOGBHS	**0,0088**	**0,0175**	**0,0147**	**0,001697645**	**100%**	**13:24**
MOGBHS-HC	0,0107	0,0207	0,0155	0,002364544	100%	13:25
MOGBHS-SA	0,0115	0,0213	0,0174	0,002086068	100%	13:24
MOGBHS-ILS	0,0102	0,0188	0,0158	0,001513986	100%	13:28
SPEA-2	0,0163	0,0337	0,0247	0,00436822	100%	13:14
SPEA-2-HC	0,0116	0,0324	0,0201	0,005081893	100%	13:01
SPEA-2-SA	0,0154	0,0343	0,0225	0,003933253	100%	13:13
SPEA-2-ILS	**0,0134**	**0,0288**	**0,0187**	**0,003381781**	**100%**	**13:11**

Tabla 5: Resultados obtenidos por todos los algoritmos en la red McTrans.

Finalmente para esta red todos los algoritmos cumplen con el criterio de calibración descrito en la sección 4.2 e incluso todos tienen un valor de GEH menor a 5 para el 100% de los enlaces, la anterior tabla muestra el estadístico GEH del mejor resultado encontrado en cada algoritmo. Con el propósito de una mejor claridad la Figura 27 ilustra los valores del estadístico GEH, antes y después del proceso de calibración de las propuestas que obtuvieron mejor promedio de NRMS de los algoritmos NSGA-II, MOGBHS, SPEA-2. El estadístico GEH se realizó con los datos obtenidos por la ejecución que obtuvo el mejor resultado de cada una de las propuestas.

La línea negra representa la condición inicial del modelo (antes de la calibración). El valor inicial de GEH era menor de 5 para aproximadamente el 50% de los enlaces. Las líneas roja, azul y verde representan la condición del modelo después de la calibración utilizando los algoritmos NSGA-II-ILS, MOGBHS y SPEA-2-ILS respectivamente. El GEH mejoró considerablemente ya que se redujo, no solo a 5 sino, a un valor menor a 1 para el 100% de los enlaces con los algoritmos NSGA-II-ILS y MOGBHS, mientras que con SPEA-2-ILS solo un enlace supera el valor de 1.

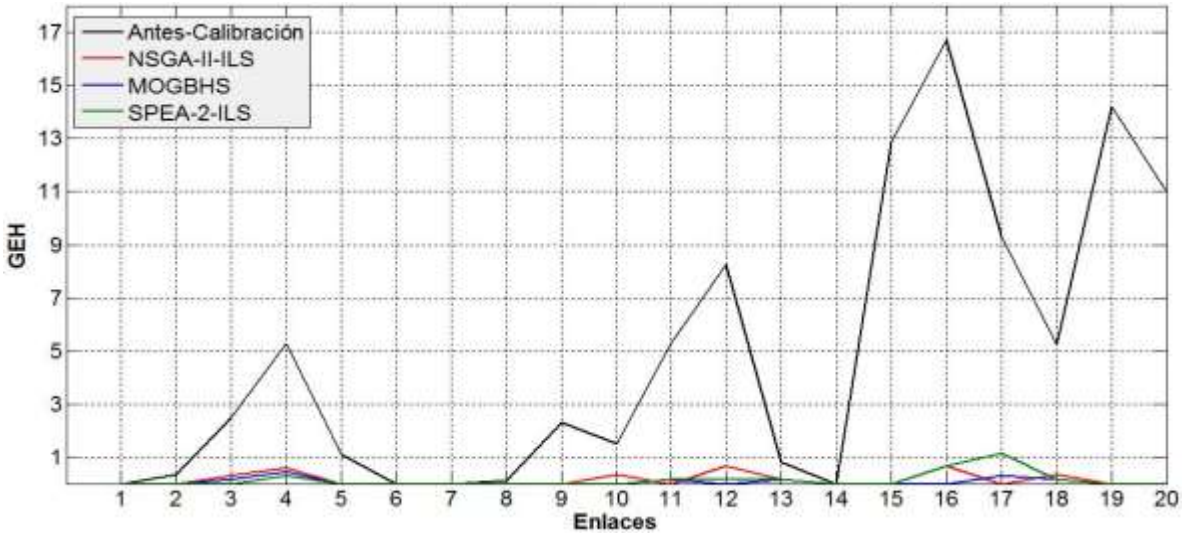

Figura 27: Estadística GEH en la red McTrans de los algoritmos NSGA-II-ILS, MOGBHS y SPEA-2-ILS

5.4.1.4 COMPARACIÓN CON GASA Y SPSA

A continuación se presenta una comparación de la propuesta que obtiene mejores resultados, referentes a NRMS, en la red de complejidad baja con los algoritmos del estado del arte GASA y SPSA.

La Figura 28 muestra el promedio de las 30 ejecuciones de los algoritmos: MOGBHS, GASA y SPSA. Se logra apreciar como el algoritmo GASA obtiene mejores resultados al inicio de la ejecución, esto dado a su estrategia de explotación sobre el mejor individuo, sin embargo el algoritmo MOGBHS converge más rápido a una óptima solución, tiene un comportamiento mucho más equilibrado y obtiene un mejor valor promedio de NRMS en comparación con los demás algoritmos. Referente al algoritmo SPSA se observa como este algoritmo no logra converger rápidamente a una buena solución y su comportamiento es el más desequilibrado en comparación con los demás. El valor promedio de NRMS para GASA y SPSA es de 0,0235 y 0,0821 lo que equivale una mejora aproximada del 92% y 71,8% respectivamente.

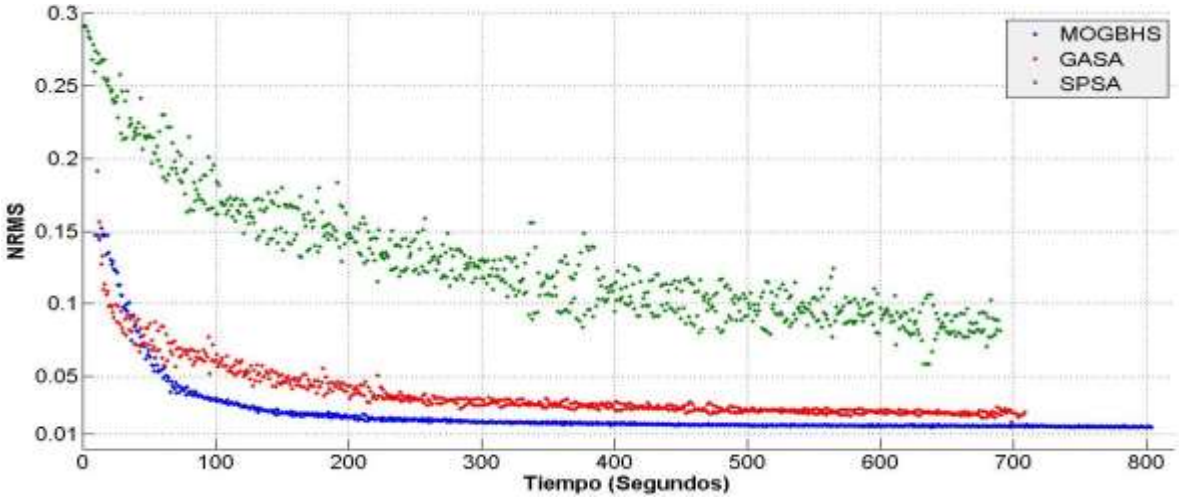

Figura 28: Gráfica NRMS vs Tiempo en la red McTrans de la propuesta ganadora con los algoritmos del estado del arte.

Por último la Tabla 6 muestra un resumen de los resultados obtenidos por los algoritmos del estado del arte, GASA y SPSA en la red de complejidad baja. Con ella se logra analizar que el tiempo de ejecución de estos algoritmos es menor que el de los algoritmos propuestos en este proyecto. Sin embargo el promedio de ejecución obtenido por el algoritmo GASA solo es superior, y por muy poco a los algoritmos de búsqueda local y al multi-objetivo SPEA-2. Finalmente, al igual que los algoritmos propuestos en esta investigación, GASA y SPSA logran cumplir con el criterio de calibración, con un valor menor a 5 para el 100% de los enlaces, esto se observa en la Tabla 6 donde se ilustra el estadístico GEH del mejor resultado encontrado en cada algoritmo.

Algoritmo	NRMS				GEH del mejor resultado	Tiempo promedio (mm:ss)
	Mejor resultado	Peor resultado	Promedio	Desviación estándar		
GASA	0,0119	0,0354	0,02359	0,005046561	100%	11:49
SPSA	0,0248	0,2163	0,08214	0,040124355	100%	11:30

Tabla 6: Resultados obtenidos por GASA y SPSA en la red McTrans.

5.4.1.5 ANÁLISIS ESTADÍSTICO

A continuación se presenta los resultados arrojados por las pruebas estadísticas de Wilcoxon y Friedman realizadas a todos los algoritmos propuestos y del estado del arte en la red de complejidad baja, con respecto a los valores de NRMS obtenidos al finalizar cada una de las 30 ejecuciones.

La Figura 29 muestra la dominancia de cada algoritmo con respecto a los demás, el punto negro representa que el algoritmo de la fila domina al algoritmo de la columna, el punto blanco indica que el algoritmo de la columna domina al de la fila y los espacios en blanco significan que no se puede determinar una dominancia. Los resultados por encima y debajo de la diagonal tienen un nivel de significancia del 0,9 y 0,95 respectivamente. De esta figura se puede analizar que el algoritmo MOGBHS es el único que domina a todos los demás algoritmos, excepto a MOGBHS-HC con quien no se puede determinar una dominancia. Por otro lado el algoritmo del estado del arte SPSA es dominado por todos los algoritmos. Así mismo el algoritmo GASA solo lograr determinar una dominancia con SPSA y este es dominado por MOGBHS y sus propuestas meméticas además de SPEA-2-HC y SPEA-2-ILS. Con los demás algoritmos propuestos en esta investigación no se logra determinar una dominancia.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)
HC (1)	-							○	○	○	○		○		○		●
SA (2)		-						○	○	○	○		○		○		●
ILS (3)			-					○	○	○	○				○		●
NSGA-II (4)				-			○	○	○	○	○		○		○		●
NSGA-II-HC (5)					-			○	○	○	○	●			○		●
NSGA-II-SA (6)						-		○	○	○	○	●			○		●
NSGA-II-ILS (7)							-	○	○	○	○	●			○		●
MOGBHS (8)	●	●	●	●	●	●	●	-		●	●	●	●	●	●	●	●
MOGBHS-HC (9)	●	●	●	●	●	●	●		-	●		●	●	●	●	●	●
MOGBHS-SA (10)	●	●	●	●	●	●	●	○	○	-	○	●	●	●		●	●
MOGBHS-ILS (11)	●	●	●	●	●	●	●	○		●	-	●	●	●	●	●	●
SPEA-2 (12)					○	○	○	○	○	○	○	-	○	○	○		●
SPEA-2-HC (13)	●	●		●				○	○	○	○	●	-	●		●	●
SPEA-2-SA (14)								○	○	○	○		○	-	○		●
SPEA-2-ILS (15)	●	●	●	●	●	●	●	○	○		○	●		●	-	●	●
GASA (16)								○	○	○	○		○		○	-	●
SPSA (17)	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	-

Figura 29: Resultados de la prueba de Wilcoxon para la red McTrans.

La Tabla 7 muestra los resultados obtenidos por todos los algoritmos en la prueba de Friedman, a cada algoritmo se le ha sido asignado un ranking, entre menor sea el valor de este ranking mejor es el algoritmo estadísticamente. Con esta tabla se ratifica que el algoritmo que presenta mejor comportamiento en este experimento es el MOGBHS, de la misma forma se observa que el último lugar es ocupado por

el algoritmo SPSA. Con respecto a los algoritmos meméticos se puede observar que las mejores propuestas son las que incluyen al algoritmo ILS como algoritmo de búsqueda local, seguido por las que incluyen al algoritmo HC. Por último con respecto a los algoritmos NSGA-II y SPEA-2 se observa como sus propuestas meméticas siempre obtienen mejores resultados que los algoritmos multi-objetivo por si solos.

ALGORITMO	RANKING
MOGBHS	2,2667
MOGBHS-ILS	3,3
MOGBHS-HC	3,3667
MOGBHS-SA	5,5333
SPEA-2-ILS	6,7333
SPEA-2-HC	8,1
NSGA-II-ILS	9,4333
NSGA-II-HC	9,6333
HC	10,2667
ILS	10,3
NSGA-II-SA	10,3
SPEA-2-SA	10,7667
SA	11,1667
GASA	11,2
NSGA-II	11,6
SPEA-2	12,2333
SPSA	16,8

Tabla 7: Resultados de la prueba de Friedman para la red McTrans.

5.4.2 RESULTADOS OBTENIDOS PARA EL MODELO RENO

5.4.2.1 RESULTADOS OBTENIDOS CON NSGA-II

A continuación se presentan los resultados obtenidos en el modelo de complejidad media por el algoritmo multi-objetivo NSGA-II y sus propuestas meméticas. Los parámetros utilizados para la ejecución son los mismos utilizados en la red de complejidad baja con la excepción de que el valor del porcentaje de mutación para el método de mutación (multi-gen) fue fijado en 5%, esto debido a la gran cantidad de genes del individuo para esta red.

La Figura 30 muestra el promedio de las 30 ejecuciones de los algoritmos: HC, NSGA-II y la propuesta memética del NSGA-II con HC denominado NSGA-II-HC. Se logra observar cómo, al igual que en la red de complejidad baja, el algoritmo HC tiende a converger rápidamente, pero este se estanca en un óptimo local, lo cual se puede ver a lo largo del tiempo, el valor promedio de NRMS para este algoritmo es igual a 0,0977 lo que equivale a una mejora de aproximadamente 56%. Por otro lado se puede apreciar que el algoritmo multi-objetivo NSGA-II y la propuesta memética NSGA-II-HC, tienden a valores similares de tiempo de ejecución y promedio de NRMS, pero el algoritmo NSGA-II presenta un comportamiento mucho

más equilibrado que el NSGA-II-HC a lo largo del tiempo. Los valores promedios de NRMS para NSGA-II y NSGA-II-HC son 0,0938 y 0,0957 lo que representa una mejora aproximada del 57,7% y 57% respectivamente.

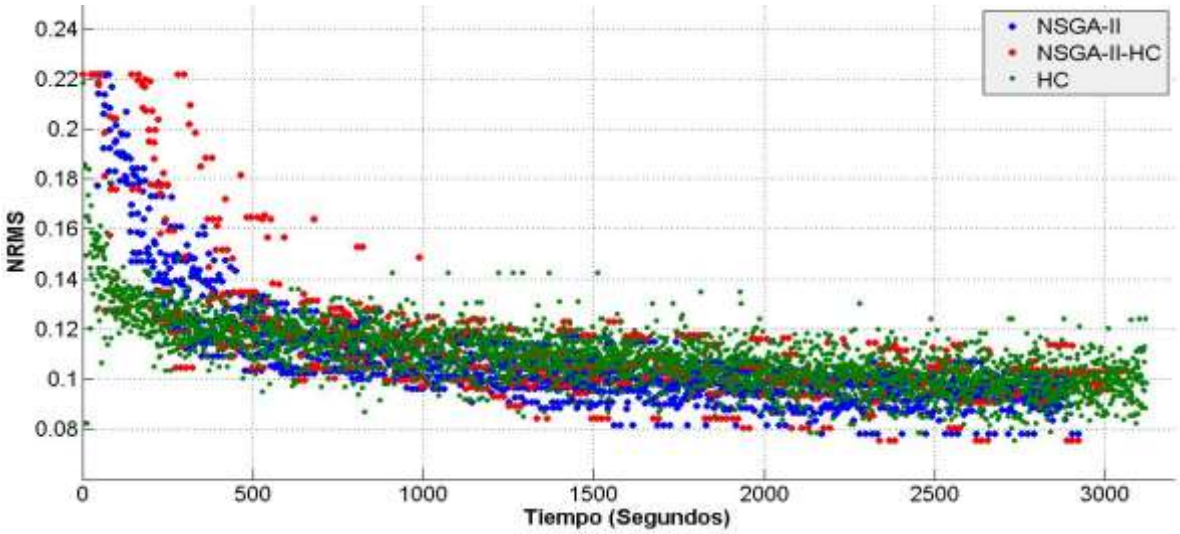

Figura 30: Gráfica NRMS vs Tiempo en la red Reno del algoritmo NSGA-II y su propuesta memética que incluye HC.

La Figura 31 muestra el promedio de las 30 ejecuciones de los algoritmos: SA, NSGA-II y la propuesta memética del NSGA-II con SA denominado NSGA-II-SA. Al igual que el HC, el algoritmo SA tiende a converger rápidamente pero se estanca en un óptimo local, su valor de NRMS promedio es de 0,1016 lo que equivale a una mejora de aproximadamente 54%. Referente a la propuesta memética NSGA-II-SA, se observa cómo éste tiende a converger más rápido que el algoritmo NSGA-II. Además se logra apreciar cómo el algoritmo NSGA-II-SA se comporta de manera más equilibrada en comparación con la propuesta memética NSGA-II-HC, esto se debe a la estrategia del SA de salir de óptimos locales. El valor de NRMS promedio para la propuesta memética es de 0,0923 lo cual corresponde a una mejora de aproximadamente el 58,4%. Finalmente el tiempo de ejecución de la propuesta memética es bastante similar al del algoritmo multi-objetivo NSGA-II.

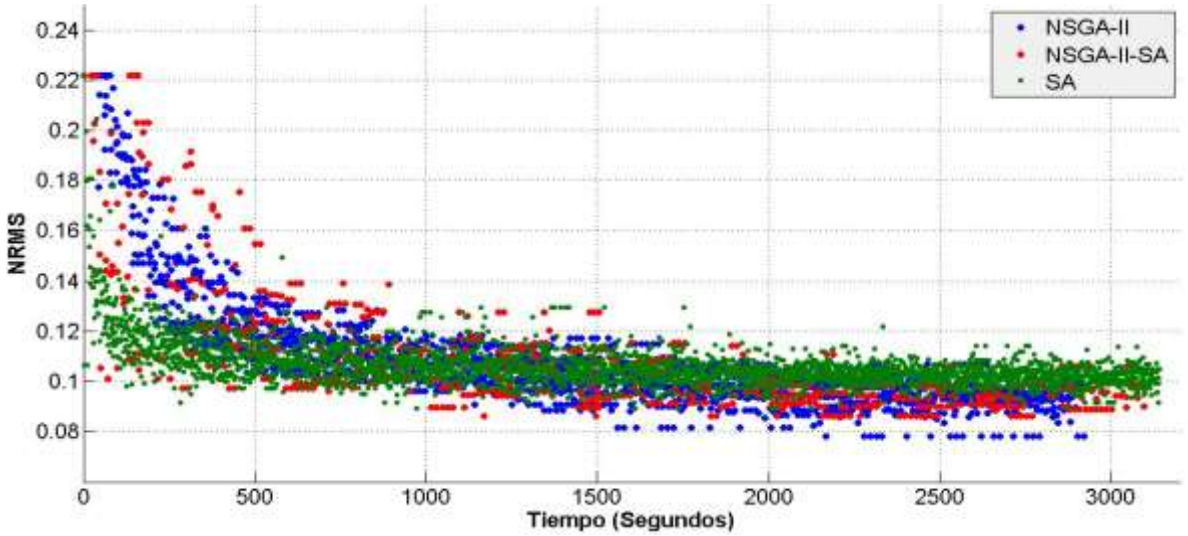

Figura 31: Gráfica NRMS vs Tiempo en la red Reno del algoritmo NSGA-II y su propuesta memética que incluye SA.

La Figura 32 muestra el promedio de las 30 ejecuciones de los algoritmos: ILS, NSGA-II y la propuesta memética del NSGA-II con ILS denominado NSGA-II-ILS. Al igual que los anteriores algoritmos de búsqueda local, el ILS tiende a converger rápidamente con la diferencia que éste no se estanca en un óptimo local y su comportamiento es más desequilibrado. El valor de NRMS promedio para este algoritmo de búsqueda local es de 0,0944 lo que equivale a una mejora de aproximadamente 57,4%. Además este algoritmo tiene un tiempo de ejecución similar a NSGA-II y NSGA-II-ILS. Con respecto a la propuesta memética se logra apreciar que esta, inicialmente se comporta de manera similar al algoritmo NSGA-II, obteniendo al final del tiempo una ligera mejora de NRMS llegando a un valor promedio de 0,0917 lo que equivale una mejora del 58,6% aproximadamente. Finalmente se observa que la propuesta memética NSGA-II-ILS se comporta de forma más equilibrada en comparación a las anteriores propuestas meméticas.

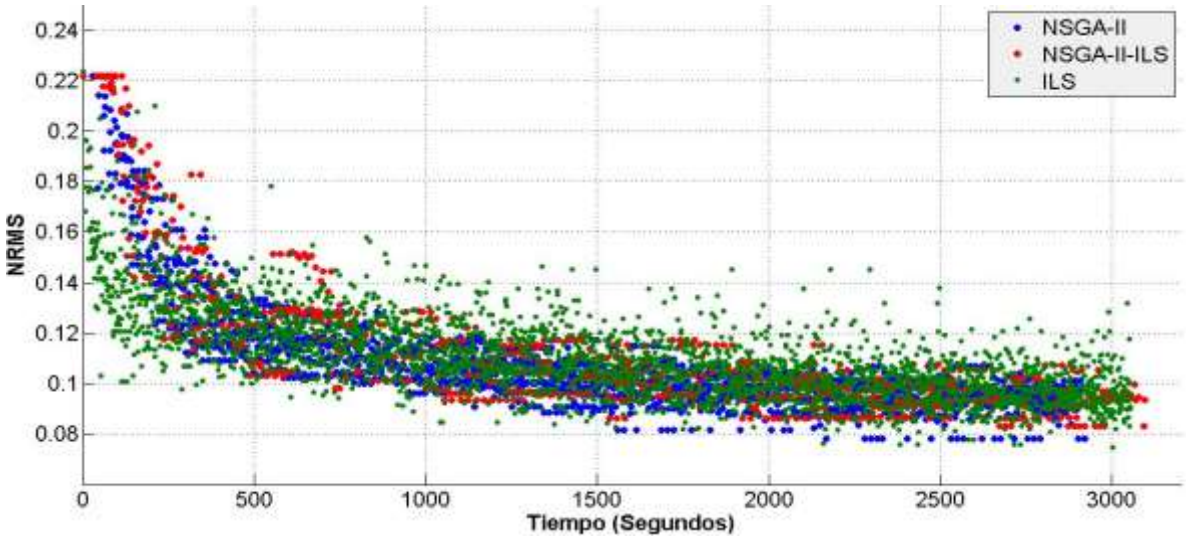

Figura 32: Gráfica NRMS vs Tiempo en la red Reno del algoritmo NSGA-II y su propuesta memética que incluye ILS.

5.4.2.2 RESULTADOS OBTENIDOS CON MOGBHS

A continuación se presentan los resultados obtenidos en el modelo de complejidad media por el algoritmo multi-objetivo MOGBHS y sus propuestas meméticas. Los parámetros utilizados para la ejecución son los mismos utilizados en la red de complejidad baja.

La Figura 33 muestra el promedio de las 30 ejecuciones de los algoritmos: HC, MOGBHS y la propuesta memética del MOGBHS con HC denominado MOGBHS-HC. De esta figura se logra analizar que el algoritmo HC es quien obtiene mejores resultados al inicio de la ejecución, esto debido a su amplia explotación. Además su tiempo de ejecución es ligeramente mayor, por otro lado, tanto el algoritmo multi-objetivo MOGBHS como la propuesta memética MOGBHS-HC mejoran a través del tiempo demostrando que gracias a la amplia exploración del espacio de búsqueda estos algoritmos evitan estancarse en óptimos locales. De estos dos algoritmos quien presenta mejor comportamiento es el algoritmo MOGBHS dado que presenta una conducta más equilibrada y obtiene un mejor resultado promedio de NRMS el cual es igual a 0,0785, lo que corresponde a una mejora aproximada del 64,6%. Mientras que el algoritmo MOGBHS-HC obtiene un valor promedio de NRMS igual a 0,0819 correspondiendo a una mejora del 63% aproximadamente. Esto indica que, al igual que en la red de complejidad baja, el algoritmo de búsqueda local HC sólo contribuye un poco a la convergencia del algoritmo memético.

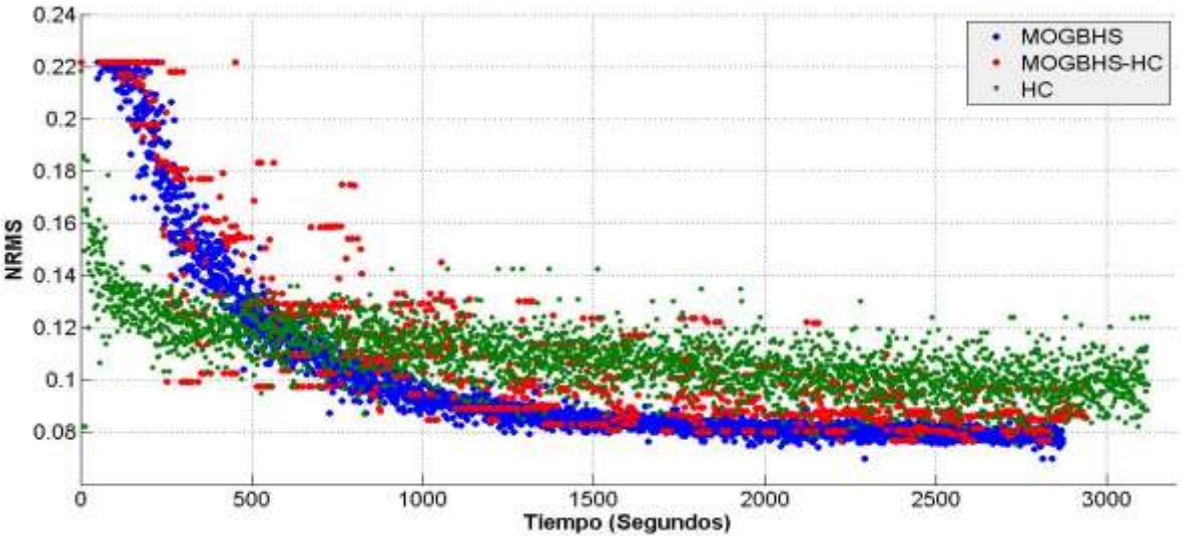

Figura 33: Gráfica NRMS vs Tiempo en la red Reno del algoritmo MOGBHS y su propuesta memética que incluye HC.

La Figura 34 ilustra el promedio de las 30 ejecuciones de los algoritmos: SA, MOGBHS y la propuesta memética del MOGBHS con SA denominado MOGBHS-SA. Se logra observar que la propuesta memética MOGBHS-SA tiende a converger un poco más rápido debido a la estrategia de explotación de los algoritmos meméticos. Por otro lado, se aprecia que su comportamiento, en comparación con la propuesta memética MOGBHS-HC es mucho más equilibrado, esto debido a la estrategia que cuenta el algoritmo SA de salir de óptimos locales. El tiempo de

ejecución de este algoritmo es bastante similar al del algoritmo MOGBHS. El valor promedio de NRMS para la propuesta memética MOGBHS-SA es de 0,0804 lo que representa una mejora del 63,7% aproximadamente. Esto indica que, al igual que HC, el algoritmo de búsqueda local SA no aporta mucho al algoritmo de búsqueda global.

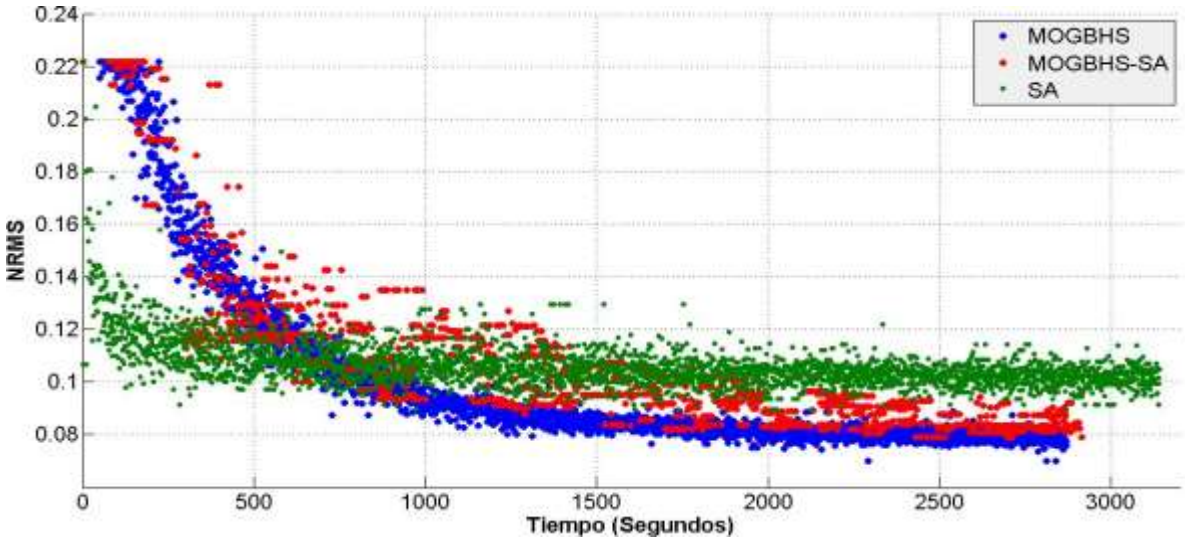

Figura 34: Gráfica NRMS vs Tiempo en la red Reno del algoritmo MOGBHS y su propuesta memética que incluye SA.

La Figura 35 ilustra el promedio de las 30 ejecuciones de los algoritmos: ILS, MOGBHS y la propuesta memética del MOGBHS con ILS denominado MOGBHS-ILS. El algoritmo de búsqueda local ILS, al igual que el algoritmo HC, es superior que el algoritmo multi-objetivo y memético multi-objetivo al inicio de la ejecución, sin embargo su tiempo de ejecución es ligeramente mayor. Por otro lado se logra apreciar que al igual que las anteriores propuestas meméticas el MOGBHS-ILS en promedio no lograr superar al algoritmo MOGBHS, lo que indica que aunque el algoritmo de búsqueda local ILS no se estanca en un óptimo local este sigue sin aportar mucho al algoritmo de búsqueda global. Se puede observar también que esta propuesta memética se comporta de manera más equilibrada a medida que el tiempo de ejecución avanza. El valor promedio de NRMS para esta propuesta es de 0,0803 lo cual corresponde a una mejora de aproximadamente 63,7%.

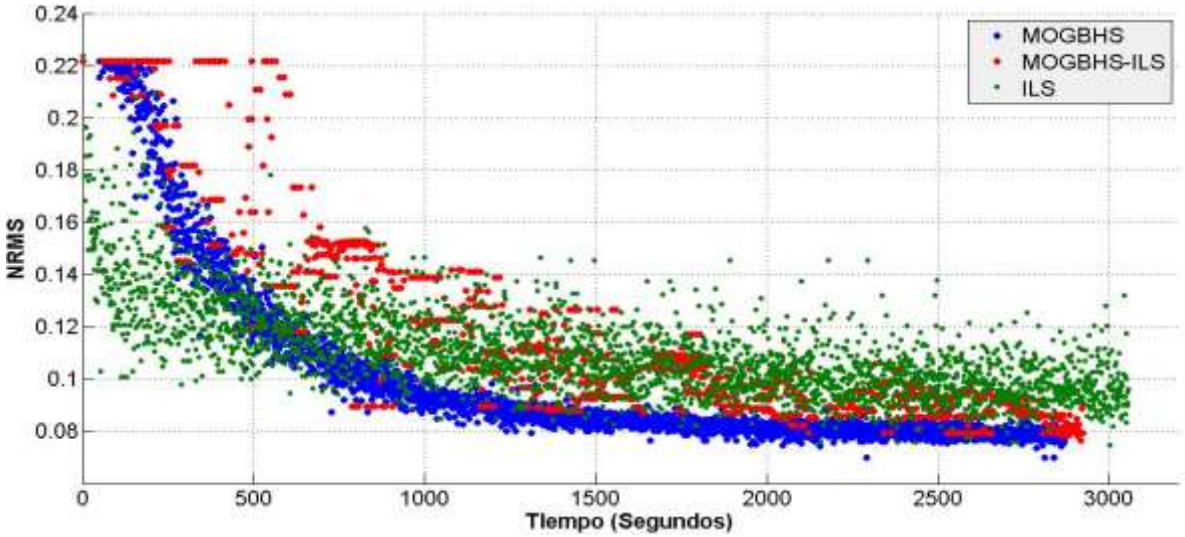

Figura 35: Gráfica NRMS vs Tiempo en la red Reno del algoritmo MOGBHS y su propuesta memética que incluye ILS.

5.4.2.3 RESULTADOS OBTENIDOS CON SPEA-2

A continuación se presentan los resultados obtenidos en el modelo de complejidad media por el algoritmo multi-objetivo SPEA-2 y sus propuestas meméticas. Similarmente a NSGA-II, los parámetros utilizados para la ejecución son los mismos utilizados en la red de complejidad baja con la excepción de que el valor del porcentaje de mutación para el método de mutación (multi-gen) fue fijado en 5%.

La Figura 36 muestra el promedio de las 30 ejecuciones de los algoritmos: HC, SPEA-2 y la propuesta memética del SPEA-2 con HC denominado SPEA-2-HC. Se puede apreciar, al igual que en NSGA-II, MOGBHS y su propuesta memética con HC, que el algoritmo HC es quien obtiene mejores resultados al inicio de la ejecución, también es observable que el algoritmo HC se encuentra a la par con el algoritmo multi-objetivo SPEA-2 pero este último no converge tan rápidamente como el anterior y su tiempo de ejecución es mucho mayor; su valor promedio de NRMS es igual a 0,0965 correspondiendo a una mejora aproximada del 56,5%. Por otro lado se puede observar que el comportamiento de la propuesta memética SPEA-2-HC es ligeramente superior a las demás, esta propuesta llega a un promedio de NRMS de 0,0917 equivalente a una mejora aproximada del 58,6%.

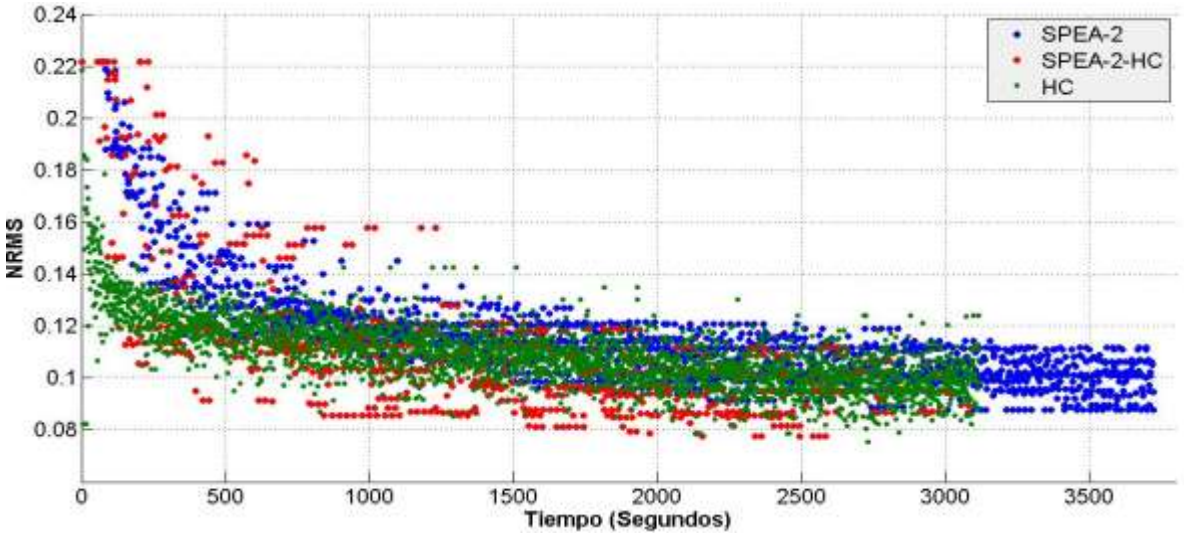

Figura 36: Gráfica NRMS vs Tiempo en la red Reno del algoritmo SPEA-2 y su propuesta memética que incluye HC.

La Figura 37 ilustra el promedio de las 30 ejecuciones de los algoritmos: SA, SPEA-2 y la propuesta memética del SPEA-2 con SA denominado SPEA-2-SA, de ella se puede apreciar que el algoritmo memético SPEA-2-SA es quien obtiene mejores resultados promedio de NRMS llegando a un valor de 0,0900 correspondiendo a una mejora aproximada del 59,4%. Además su tiempo de ejecución es el menor de los tres y presenta un comportamiento mucho más equilibrado que el algoritmo SPEA-2-HC.

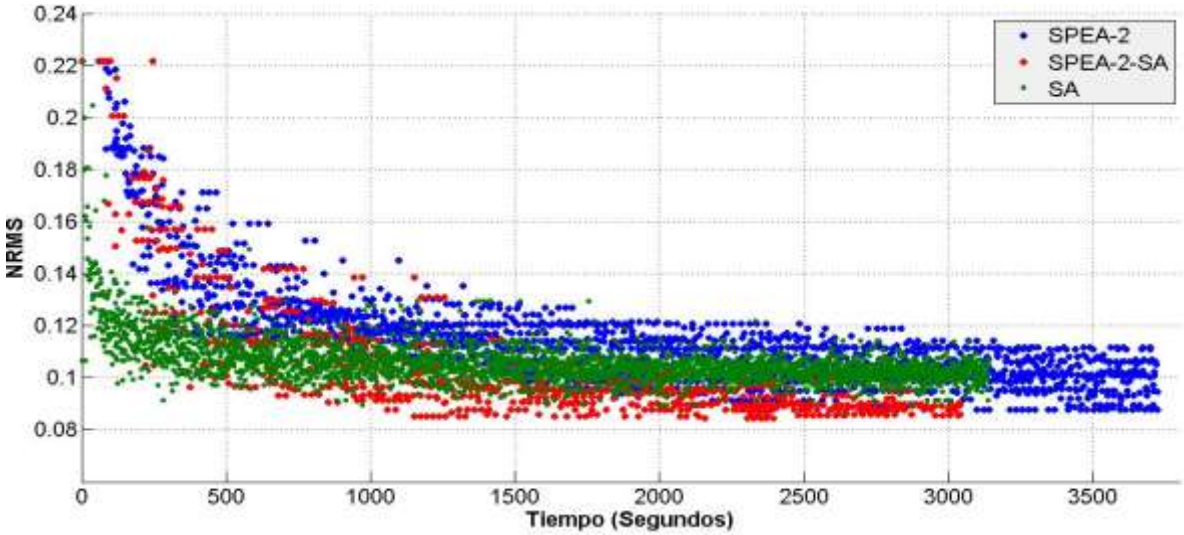

Figura 37: Gráfica NRMS vs Tiempo en la red Reno del algoritmo SPEA-2 y su propuesta memética que incluye SA.

La Figura 38 muestra el promedio de las 30 ejecuciones de los algoritmos: ILS, SPEA-2 y la propuesta memética del SPEA-2 con ILS denominado SPEA-2-ILS, de ella se puede apreciar que, al igual que en la red de complejidad baja, el algoritmo de búsqueda local obtiene un mejor valor promedio de NRMS que el algoritmo multi-objetivo. Con referencia a la propuesta memética SPEA-2-ILS, esta es quien obtiene

mejores resultados promedio de NRMS llegando a un valor de 0,0900 correspondiendo a una mejora aproximada del 59,4%. Además su tiempo de ejecución es mucho menor en comparación con SPEA-2.

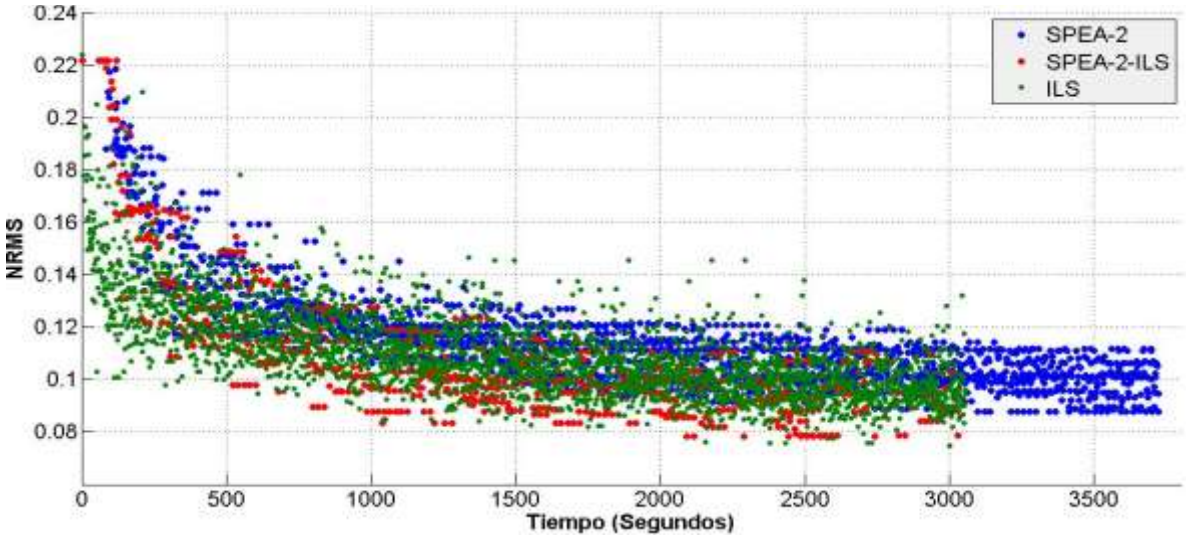

Figura 38: Gráfica NRMS vs Tiempo en la red Reno del algoritmo SPEA-2 y su propuesta memética que incluye ILS.

La Tabla 8 muestra el resumen de los resultados obtenidos por todos los algoritmos en la red de complejidad media. Se logra apreciar que los tiempos de ejecución son muy similares en todos los algoritmos con excepción del algoritmo multi-objetivo SPEA-2. Referente a las propuestas meméticas de los algoritmos NSGA-II y SPEA-2 se observa que, al igual que en la red de complejidad baja, éstas siempre logran un mejor valor promedio de NRMS, en comparación a sus multi-objetivo, y en el caso del algoritmo SPEA-2, logra disminuir significativamente el tiempo de ejecución. Con respecto al algoritmo MOGBHS, se aprecia cómo, al igual que en el anterior experimento, el mejor comportamiento se logra solo con el algoritmo multi-objetivo, sin la inclusión de un algoritmo de búsqueda local. Finalmente se observa que la propuesta memética que logra mejores resultados promedio de NRMS para los algoritmos multi-objetivo NSGA-II y MOGBHS es la que incluye al algoritmo ILS mientras que para SPEA-2 hay un empate entre la propuesta que incluye ILS y la que incluye SA, la propuesta memética que obtiene mejor valor de desviación estándar es la que incluye al algoritmo SA.

Algoritmo	NRMS				GEH del mejor resultado	Tiempo promedio (mm:ss)
	Mejor resultado	Peor resultado	Promedio	Desviación estándar		
HC	0,0754	0,1239	0,0977	0,011790708	100%	51:50
SA	0,0894	0,1142	0,1016	0,006022747	95%	52:20
ILS	0,0745	0,1281	0,0944	0,013749052	100%	50:55
NSGA-II	0,0756	0,1011	0,0938	0,005664098	100%	48:40
NSGA-II-HC	0,0754	0,1179	0,0957	0,009251681	100%	51:18
NSGA-II-SA	0,0808	0,1033	0,0923	0,004210363	100%	51:37
NSGA-II-ILS	**0,0792**	**0,1054**	**0,0917**	**0,006710928**	**100%**	**51:35**
MOGBHS	**0,0700**	**0,0873**	**0,0785**	**0,00388558**	**100%**	**47:50**
MOGBHS-HC	0,0719	0,0943	0,0819	0,005231409	100%	48:57

MOGBHS-SA	0,0743	0,0889	0,0804	0,003097676	100%	48:33
MOGBHS-ILS	0,0675	0,0917	0,0803	0,00516669	100%	48:43
SPEA-2	0,0864	0,1116	0,0965	0,007005492	100%	62:04
SPEA-2-HC	0,0759	0,1208	0,0917	0,010539557	100%	51:21
SPEA-2-SA	0,0845	0,0990	0,0900	0,00401699	100%	50:42
SPEA-2-ILS	**0,0754**	**0,1091**	**0,0900**	**0,007914779**	**100%**	**50:30**

Tabla 8: Resultados obtenidos por todos los algoritmos en la red Reno.

Para esta red todos los algoritmos cumplen con el criterio de calibración descrito en la sección 4.2 e incluso todos los algoritmos, con excepción de SA, logran un valor de GEH menor a 5 para el 100% de los enlaces, la anterior tabla muestra el estadístico GEH del mejor resultado encontrado en cada algoritmo. Con la intención de una mejor claridad la Figura 39 ilustra los valores del estadístico GEH, antes y después del proceso de calibración de las propuestas que obtuvieron mejor promedio de NRMS de los algoritmos NSGA-II, MOGBHS y SPEA-2. El estadístico GEH se realizó con los datos obtenidos por la ejecución que obtuvo el mejor resultado de cada una de las propuestas. La línea negra representa la condición inicial del modelo (antes de la calibración). El valor inicial de GEH era menor de 5 para aproximadamente el 47% de los enlaces. Las líneas roja, azul y verde representan la condición del modelo después de la calibración utilizando los algoritmos NSGA-II-ILS, MOGBHS y SPEA-2-ILS respectivamente. El GEH mejoró considerablemente ya que se obtiene un valor menor a 5 para el 100% de los enlaces.

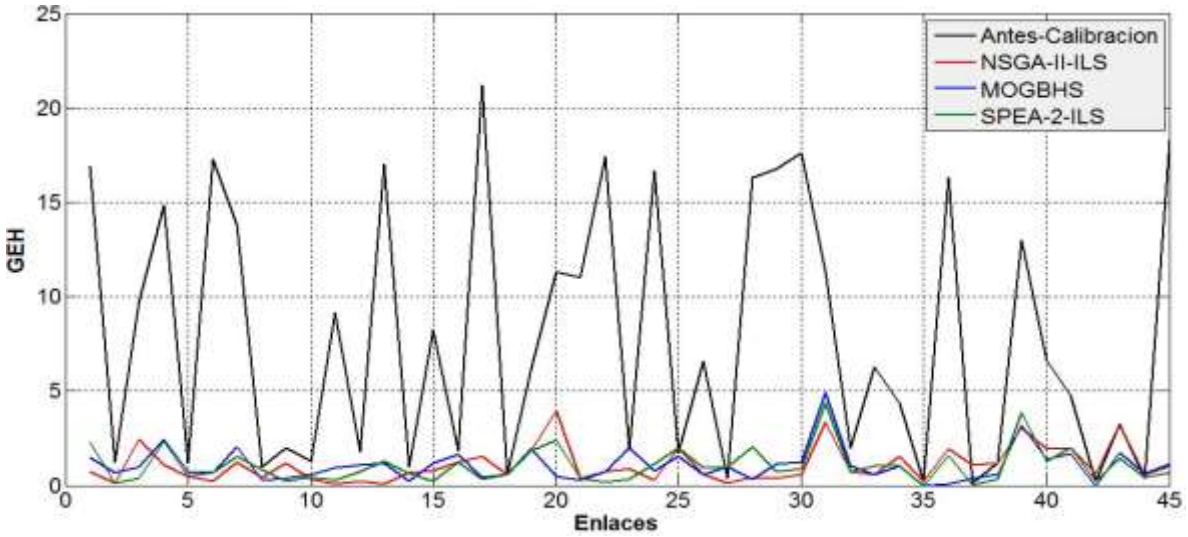

Figura 39: Estadística GEH en la red Reno de los algoritmos NSGA-II-ILS, MOGBHS y SPEA-2-ILS.

5.4.2.4 COMPARACIÓN CON GASA Y SPSA

A continuación se presenta una comparación de la propuesta que obtiene mejores resultados en la red de complejidad media con los algoritmos GASA y SPSA.

La Figura 40 muestra el promedio de las 30 ejecuciones de los algoritmos: MOGBHS, GASA y SPSA. Se logra apreciar cómo, al igual que en el anterior

experimento, el algoritmo GASA obtiene mejores resultados al inicio de la ejecución, esto dado a su estrategia de explotación sobre el mejor individuo, sin embargo el algoritmo MOGBHS, tiene un comportamiento más equilibrado y obtiene un mejor valor promedio de NRMS que los demás algoritmos. Referente al algoritmo SPSA se observa como este algoritmo no logra obtener buenos resultados de NRMS como los anteriores algoritmos y su comportamiento es el más desequilibrado en comparación con los demás. El valor promedio de NRMS para GASA y SPSA es de 0,0862 y 0,1533 lo que equivale una mejora aproximada del 61% y 31,2% respectivamente.

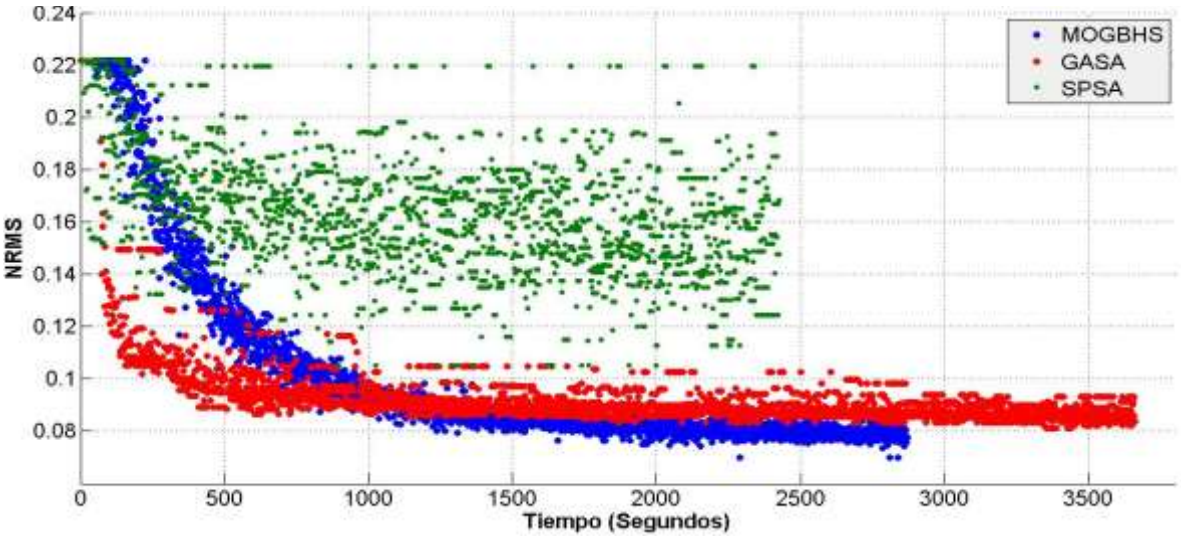

Figura 40: Gráfica NRMS vs Tiempo en la red Reno de la propuesta ganadora con los algoritmos del estado del arte.

Por último la Tabla 9 muestra un resumen de los resultados obtenidos por los algoritmos del estado del arte, GASA y SPSA en la red de complejidad media. Con ella se logra analizar que el tiempo de ejecución de SPSA es menor que el de los algoritmos propuestos en esta investigación (estos tiempos son expuestos en la Tabla 8) sin embargo su promedio de ejecución es el más alto de todos. Referente a GASA el tiempo de ejecución de este es mayor que todos los algoritmos propuestos excepto al multi-objetivo SPEA-2; el promedio de NRMS de GASA solo es superado por el algoritmo multi-objetivo MOGBHS y sus propuestas meméticas. Por último se observa el estadístico GEH del mejor resultado encontrado en cada algoritmo, se muestra como ambos algoritmos cumplen con el criterio de calibración, se destaca como GASA logra un valor menor a 5 para el 100% de los enlaces.

Algoritmo	NRMS				GEH del mejor resultado	Tiempo promedio (mm:ss)
	Mejor resultado	Peor resultado	Promedio	Desviación estándar		
GASA	0,0795	0,0932	0,0862	0,003521316	100%	61:00
SPSA	0,1051	0,2196	0,1526	0,025917078	91%	40:26

Tabla 9: Resultados obtenidos por GASA y SPSA en la red Reno.

5.4.2.5 ANÁLISIS ESTADÍSTICO

A continuación se presenta los resultados arrojados por las pruebas estadísticas de Wilcoxon y Friedman realizadas a todos los algoritmos propuestos y del estado del arte en la red de complejidad media, con respecto a los valores de NRMS obtenidos al finalizar cada una de las 30 ejecuciones.

La Figura 41 muestra la dominancia de cada algoritmo con respecto a los demás. De esta figura se puede analizar que el algoritmo MOGBHS es el único que domina a todos los demás algoritmos, excepto a MOGBHS-SA con quien no se puede determinar una dominancia. Por otro lado, al igual que el anterior experimento, el algoritmo del estado del arte SPSA es dominado por todos los algoritmos. Así mismo, el algoritmo GASA domina a todos los algoritmos de búsqueda local, a NSGA-II y todas sus propuestas meméticas y a SPEA-2 y sus propuestas meméticas SPEA-2-HC, SPEA-2-SA mientras que con SPEA-2-ILS no se puede determinar dominancia. Finalmente el algoritmo GASA es dominado por MOGBHS y sus propuestas meméticas.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)
HC (1)	-			○		○	○	○	○	○	○			○	○	○	●
SA (2)		-		○	○	○	○	○	○	○	○		○	○	○	○	●
ILS (3)			-					○	○	○	○			○	○	○	●
NSGA-II (4)		●		-				○	○	○	○			○		○	●
NSGA-II-HC (5)		●			-			○	○	○	○			○		○	●
NSGA-II-SA (6)	●	●				-		○	○	○	○					○	●
NSGA-II-ILS (7)		●					-	○	○	○	○	●				○	●
MOGBHS (8)	●	●	●	●	●	●	●	-	●		●	●	●	●	●	●	●
MOGBHS-HC (9)	●	●	●	●	●	●	●	○	-			●	●	●	●	●	●
MOGBHS-SA (10)	●	●	●	●	●	●	●			-		●	●	●	●	●	●
MOGBHS-ILS (11)	●	●	●	●	●	●	●				-	●	●	●	●	●	●
SPEA-2 (12)								○	○	○	○	-	○	○	○	○	●
SPEA-2-HC (13)		●						○	○	○	○		-			○	●
SPEA-2-SA (14)	●	●			●			○	○	○	○	●		-		○	●
SPEA-2-ILS (15)	●	●						○		○	○	●			-		●
GASA (16)	●	●	●	●	●	●	●	○	○	○	○	●	●	●		-	●
SPSA (17)	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	-

Figura 41: Resultados de la prueba de Wilcoxon para la red Reno.

La Tabla 10 muestra los resultados obtenidos por todos los algoritmos en la prueba de Friedman. Similarmente al anterior experimento esta tabla ratifica que el algoritmo que presenta mejor comportamiento es el MOGBHS, de la misma forma se observa que el último lugar es ocupado por el algoritmo SPSA. Con respecto a los algoritmos meméticos de NSGA-II y SPEA-2 se puede observar que las mejores propuestas son las que incluyen al algoritmo ILS como algoritmo de búsqueda local, seguido muy de cerca por las que incluyen al algoritmo SA. Por último se destaca como el algoritmo GASA logra obtener un mejor ranking con respecto al anterior experimento.

ALGORITMO	RANKING
MOGBHS	2,2
MOGBHS-SA	3,0667
MOGBHS-ILS	3,6667

MOGBHS-HC	4,6
GASA	6,6
SPEA-2-ILS	8,2
SPEA-2-SA	8,4
NSGA-II-ILS	9,2667
NSGA-II-SA	9,6667
SPEA-2-HC	10
NSGA-II	10,3333
ILS	10,9333
NSGA-II-HC	11
SPEA-2	11,9333
HC	12,3333
SA	13,9333
SPSA	16,8667

Tabla 10: Resultados de la prueba de Friedman para la red Reno.

5.4.3 RESULTADOS OBTENIDOS PARA EL MODELO I-75

5.4.3.1 RESULTADOS OBTENIDOS CON NSGA-II

A continuación se presentan los resultados obtenidos en el modelo de complejidad alta por el algoritmo multi-objetivo NSGA-II y sus propuestas meméticas. Los parámetros utilizados para la ejecución son los mismos utilizados en la red de complejidad media.

La Figura 42 muestra el promedio de las 30 ejecuciones de los algoritmos: HC, NSGA-II y la propuesta memética del NSGA-II con HC denominado NSGA-II-HC. Se logra observar cómo, al igual que en los anteriores experimentos, el algoritmo HC tiende a converger rápidamente, pero este se estanca en un óptimo local, lo cual se puede ver a lo largo del tiempo, el valor promedio de NRMS para este algoritmo es igual a 0,1080 lo que equivale a una mejora de aproximadamente 67,4%. Por otro lado se puede apreciar que el algoritmo multi-objetivo NSGA-II y la propuesta memética NSGA-II-HC, tienden a valores similares de tiempo de ejecución y promedio de NRMS, pero el algoritmo NSGA-II-HC presenta un comportamiento un poco más equilibrado que NSGA-II a lo largo del tiempo. Los valores promedios de NRMS para NSGA-II y NSGA-II-HC son 0,1029 y 0,1018 lo que representa una mejora aproximada del 69% y 69,3% respectivamente. Finalmente el tiempo de ejecución de la propuesta memética es mucho menor que el del algoritmo multi-objetivo NSGA-II.

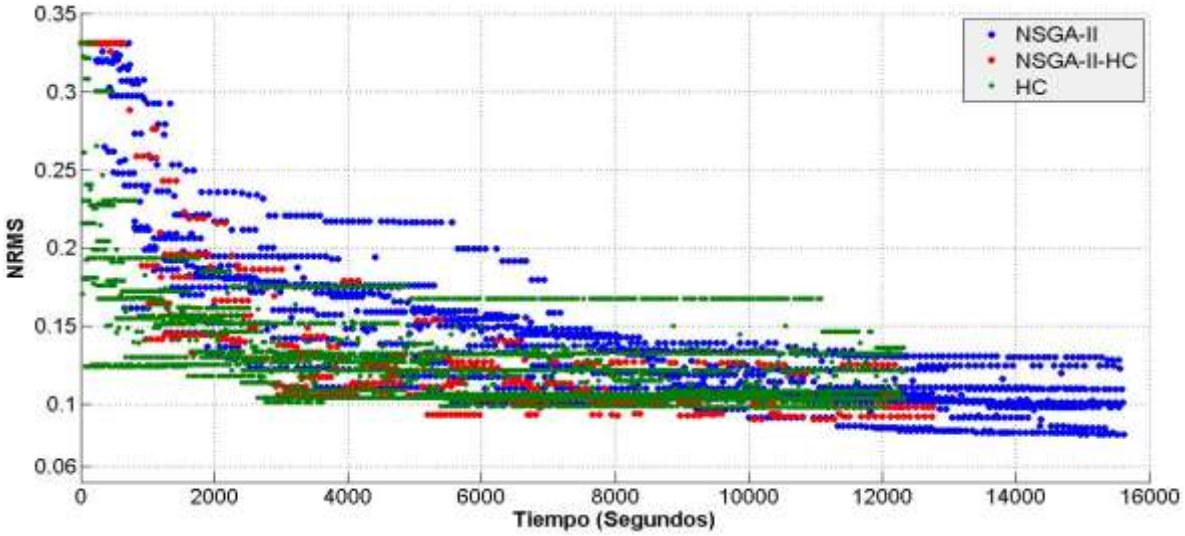

Figura 42: Gráfica NRMS vs Tiempo en la red I-75 del algoritmo NSGA-II y su propuesta memética que incluye HC.

La Figura 43 muestra el promedio de las 30 ejecuciones de los algoritmos: SA, NSGA-II y la propuesta memética del NSGA-II con SA denominado NSGA-II-SA. A diferencia de los anteriores experimentos, en este modelo el algoritmo SA no tiende a converger rápidamente a una buena solución, este algoritmo se estanca rápidamente en un óptimo local, su valor de NRMS promedio es de 0,2766 lo que equivale a una mejora de aproximadamente 16,6%. Es por tanto que la propuesta memética NSGA-II-SA, no mejora con respecto al algoritmo NSGA-II, el promedio de NRMS para el algoritmo memético es de 0,1293 lo cual corresponde a una mejora de aproximadamente el 61%.

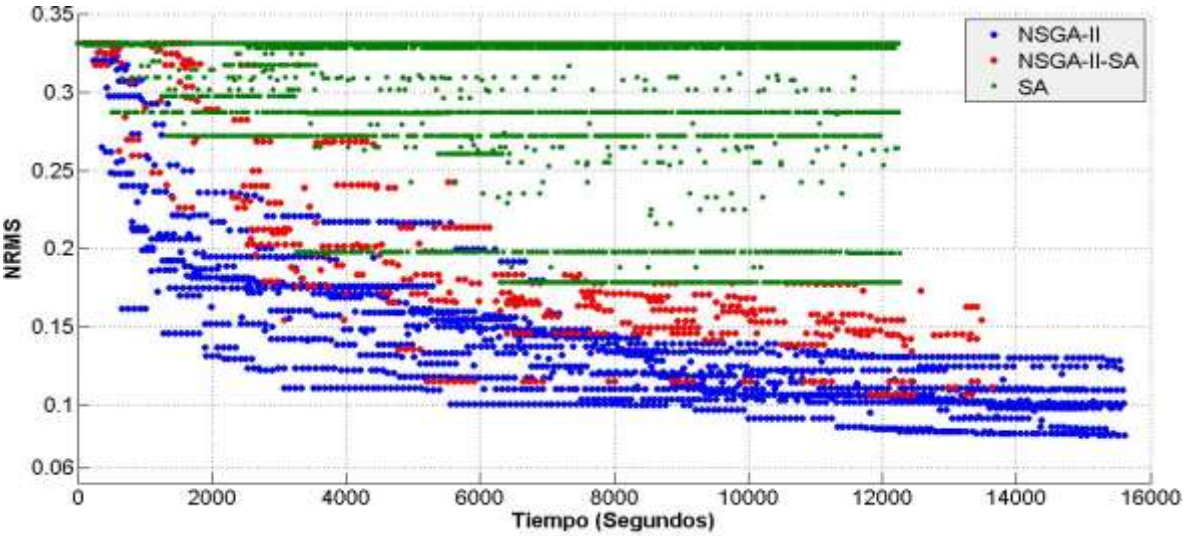

Figura 43: Gráfica NRMS vs Tiempo en la red I-75 del algoritmo NSGA-II y su propuesta memética que incluye SA.

La Figura 44 muestra el promedio de las 30 ejecuciones de los algoritmos: ILS, NSGA-II y la propuesta memética del NSGA-II con ILS denominado NSGA-II-ILS. El algoritmo ILS tiende a converger rápidamente pero, a diferencia de HC, este no se

estanca en un óptimo local y su comportamiento es más desequilibrado, el valor de NRMS promedio para este algoritmo de búsqueda local es de 0,1069 lo que equivale a una mejora de aproximadamente 67,8%, además este algoritmo tiene un tiempo de ejecución similar a NSGA-II-ILS. Correspondiente a la propuesta memética se logra apreciar como esta, se comporta de manera más desequilibrada en comparación a la propuesta memética NSGA-II-HC. Si bien el algoritmo NSGA-II-ILS reduce significativamente el tiempo de ejecución, en comparación con NSGA-II, este algoritmo no logra mejoras significativas de NRMS, su valor promedio es de 0,1122 lo que equivale una mejora del 66,2% aproximadamente.

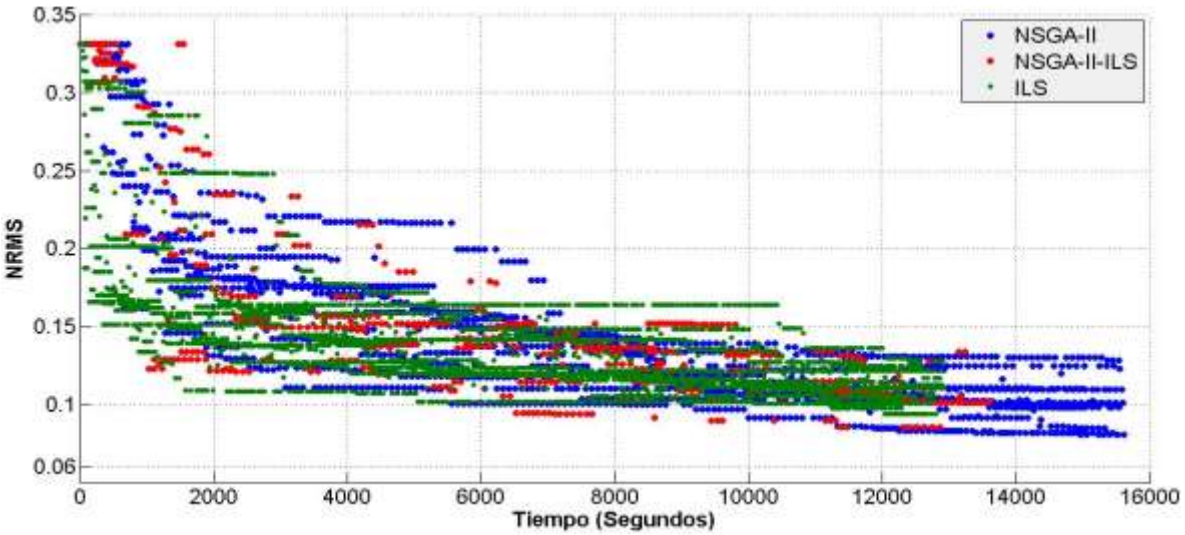

Figura 44: Gráfica NRMS vs Tiempo en la red I-75 del algoritmo NSGA-II y su propuesta memética que incluye ILS.

5.4.3.2 RESULTADOS OBTENIDOS CON MOGBHS

A continuación se presentan los resultados obtenidos en el modelo de complejidad alta por el algoritmo multi-objetivo MOGBHS y sus propuestas meméticas. Los parámetros utilizados para la ejecución son los mismos utilizados en la red de complejidad baja y media.

La Figura 45 muestra el promedio de las 30 ejecuciones de los algoritmos: HC, MOGBHS y la propuesta memética del MOGBHS con HC denominado MOGBHS-HC. De esta figura se logra analizar que el algoritmo HC es quien obtiene mejores resultados al inicio de la ejecución, esto debido a su amplia explotación, por otro lado la propuesta memética MOGBHS-HC tiende a converger más rápido que el algoritmo multi-objetivo MOGBHS, estos dos algoritmos mejoran a través del tiempo demostrando que gracias a la amplia exploración del espacio de búsqueda estos algoritmos evitan estancarse en óptimos locales. De estos dos algoritmos quien presenta mejor comportamiento es el algoritmo MOGBHS dado que presenta una conducta más equilibrada y obtiene un mejor resultado promedio de NRMS el cual es igual a 0,0746 lo que corresponde a una mejora aproximada del 77,5%. Mientras que el algoritmo MOGBHS-HC obtiene un valor promedio de NRMS igual a 0,0872 correspondiendo a una mejora del 73,7% aproximadamente. Esto indica que el

método de búsqueda local HC sólo contribuye a la rápida convergencia del algoritmo memético, esta contribución es mucho mayor, en comparación con los anteriores experimentos.

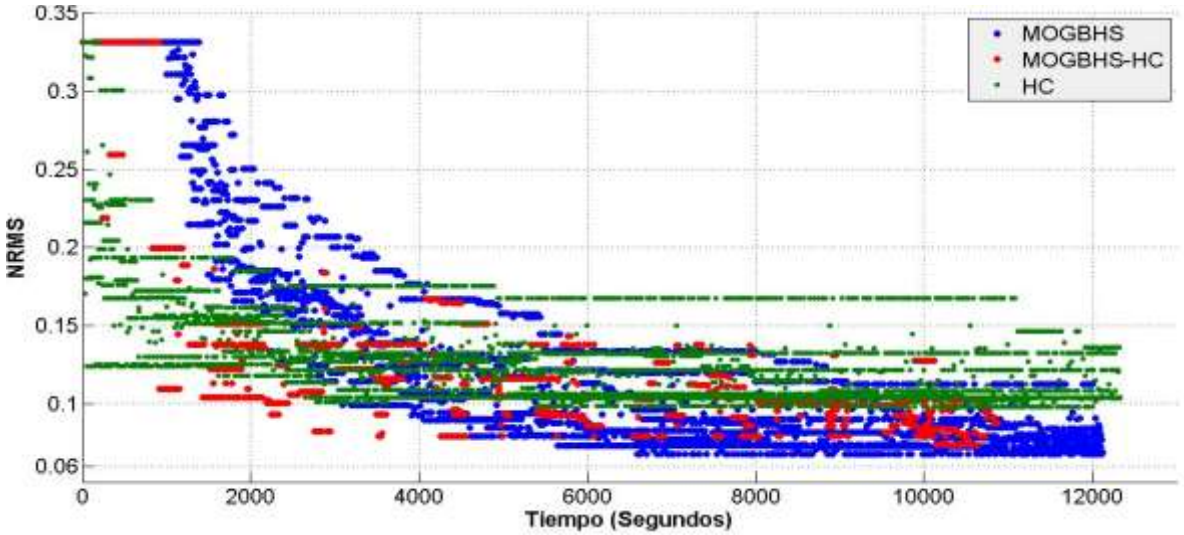

Figura 45: Gráfica NRMS vs Tiempo en la red I-75 del algoritmo MOGBHS y su propuesta memética que incluye HC.

La Figura 46 ilustra el promedio de las 30 ejecuciones de los algoritmos: SA, MOGBHS y la propuesta memética del MOGBHS con SA denominado MOGBHS-SA. Se logra observar que la propuesta memética MOGBHS-SA presenta un comportamiento bastante desequilibrado, si bien hay puntos donde esta propuesta obtiene mejores valores de NRMS que el algoritmo MOGBHS, no se puede concluir que esta propuesta es superior en ningún momento. El tiempo de ejecución de este algoritmo es menor en comparación a los algoritmos MOGBHS y SA. El valor promedio de NRMS para la propuesta memética MOGBHS-SA es de 0,0969 lo que representa una mejora del 70,8% aproximadamente.

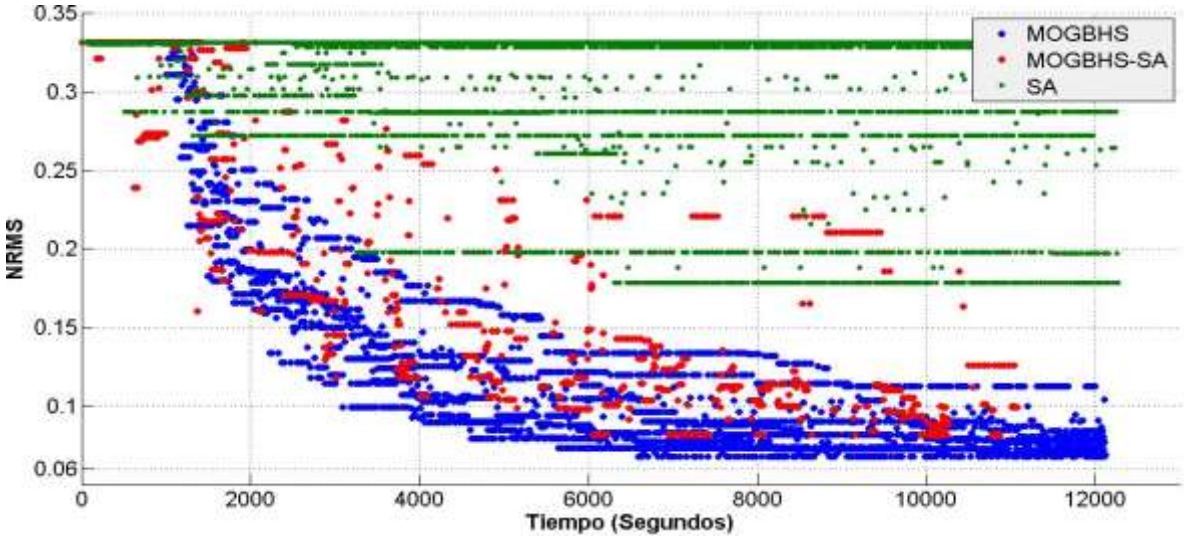

Figura 46: Gráfica NRMS vs Tiempo en la red I-75 del algoritmo MOGBHS y su propuesta memética que incluye SA.

La Figura 47 ilustra el promedio de las 30 ejecuciones de los algoritmos: ILS, MOGBHS y la propuesta memética del MOGBHS con ILS denominado MOGBHS-ILS. El algoritmo de búsqueda local ILS, al igual que el algoritmo HC, es superior que el algoritmo multi-objetivo y memético multi-objetivo al inicio de la ejecución, sin embargo su tiempo de ejecución es mayor. Por otro lado se logra apreciar que a diferencia de los anteriores experimentos, MOGBHS-ILS logra converger más rápido que el algoritmo MOGBHS, pero éste último tiene un comportamiento mucho más equilibrado y logra mejor valor promedio de NRMS, esto indica que aunque el algoritmo de búsqueda local ILS no se estanca en un óptimo local este sigue sin aportar mucho al algoritmo de búsqueda global. Se puede observar también como esta propuesta memética se comporta de manera más equilibrada a medida que el tiempo de ejecución avanza. El valor promedio de NRMS para esta propuesta es de 0,0803 lo cual corresponde a una mejora de aproximadamente 75,9%. Esto indica que, al igual que HC, el método de búsqueda ILS sólo contribuye a la rápida convergencia del algoritmo memético.

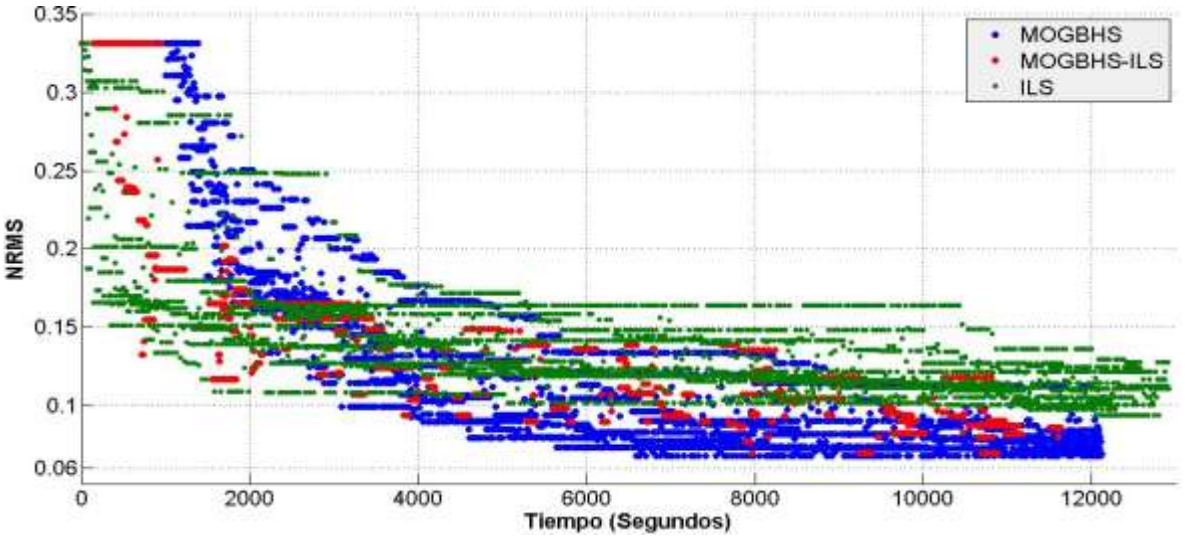

Figura 47: Gráfica NRMS vs Tiempo en la red I-75 del algoritmo MOGBHS y su propuesta memética que incluye ILS.

5.4.3.3 RESULTADOS OBTENIDOS CON SPEA-2

A continuación se presentan los resultados obtenidos en el modelo de complejidad alta por el algoritmo multi-objetivo SPEA-2 y sus propuestas meméticas. Los parámetros utilizados para la ejecución son los mismos utilizados en la red de complejidad media.

La Figura 48 muestra el promedio de las 30 ejecuciones de los algoritmos: HC, SPEA-2 y la propuesta memética del SPEA-2 con HC denominado SPEA-2-HC. Se puede apreciar, al igual que en NSGA-II, MOGBHS y su propuesta memética con HC, que el algoritmo HC es quien obtiene mejores resultados al inicio de la ejecución. También se observa que el algoritmo HC se encuentra a la par con el algoritmo multi-objetivo SPEA-2, pero este último no converge tan rápidamente como el anterior y su tiempo de ejecución es mucho mayor; su valor promedio de

NRMS es 0,1050 correspondiendo a una mejora aproximada del 68,3%. Por otro lado se puede observar que el comportamiento de la propuesta memética SPEA-2-HC es ligeramente superior a las demás, esta propuesta llega a un promedio de NRMS de 0,0957 equivalente a una mejora aproximada del 71,1%.

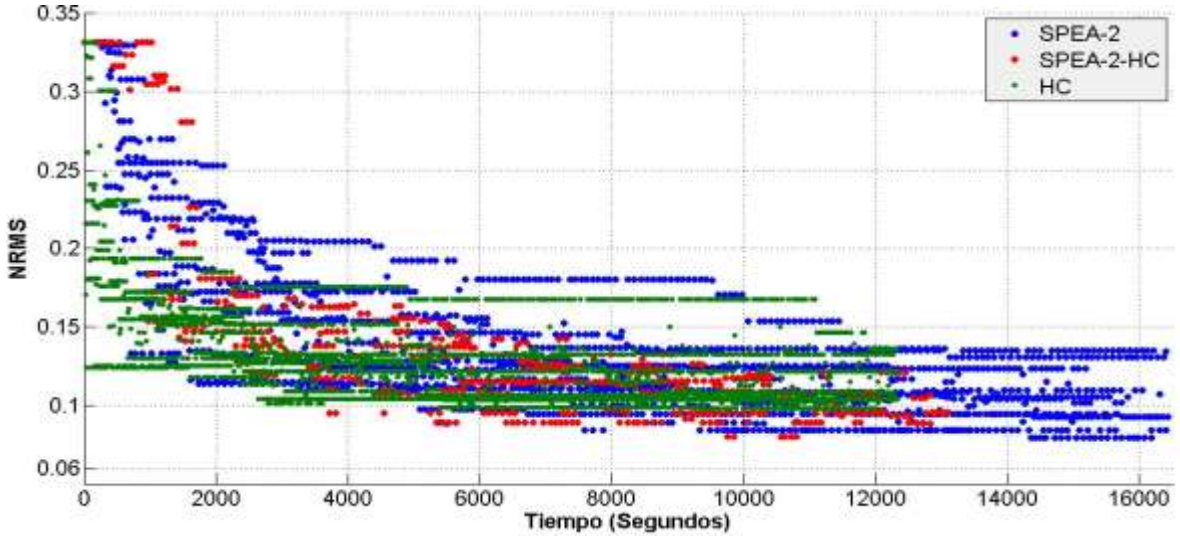

Figura 48: Gráfica NRMS vs Tiempo en la red I-75 del algoritmo SPEA-2 y su propuesta memética que incluye HC.

La Figura 49 ilustra el promedio de las 30 ejecuciones de los algoritmos: SA, SPEA-2 y la propuesta memética del SPEA-2 con SA denominado SPEA-2-SA, de ella se puede apreciar que el algoritmo memético SPEA-2-SA no mejora con respecto a SPEA-2, esto debido a que el método de búsqueda local SA no presenta un buen comportamiento en este experimento, el promedio de NRMS para la propuesta memética es de 0,1348 correspondiendo a una mejora aproximada del 59,3%.

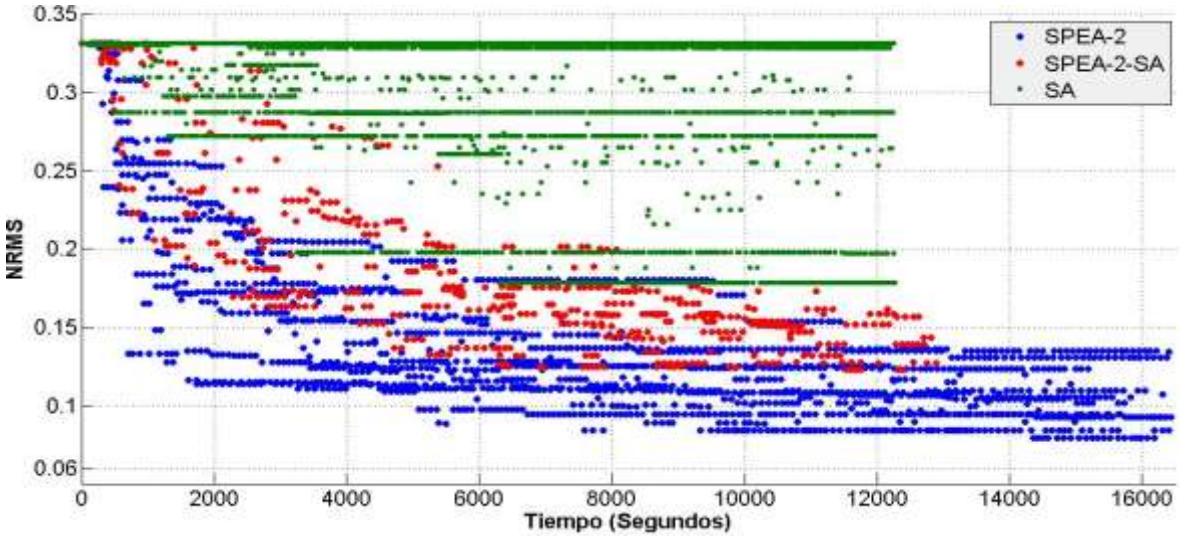

Figura 49: Gráfica NRMS vs Tiempo en la red I-75 del algoritmo SPEA-2 y su propuesta memética que incluye SA.

La Figura 50 muestra el promedio de las 30 ejecuciones de los algoritmos: ILS, SPEA-2 y la propuesta memética del SPEA-2 con ILS denominado SPEA-2-ILS, de

ella se puede apreciar que, al igual que en los anteriores experimentos, el algoritmo de búsqueda local obtiene un mejor valor promedio de NRMS que el algoritmo multi-objetivo y su tiempo de ejecución es mucho menor. Con referencia a la propuesta memética SPEA-2-ILS, esta obtiene un valor promedio de NRMS similar al de SPEA-2, llegando a un valor de 0,1079 correspondiendo a una mejora aproximada del 67,5%. Además su tiempo de ejecución es mucho menor en comparación con SPEA-2.

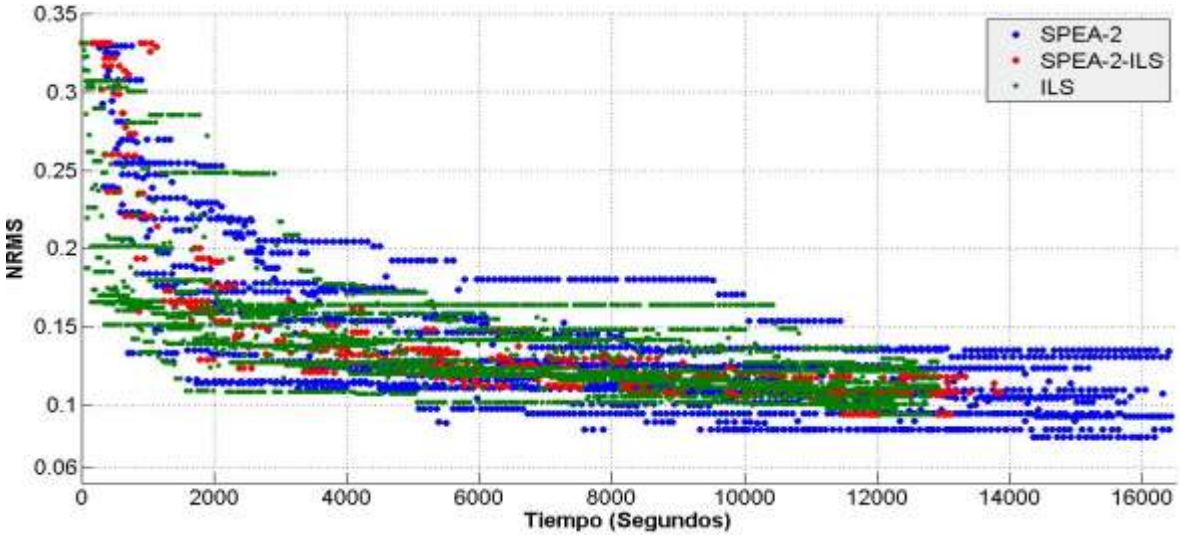

Figura 50: Gráfica NRMS vs Tiempo en la red I-75 del algoritmo SPEA-2 y su propuesta memética que incluye ILS.

La Tabla 11 muestra el resumen de los resultados obtenidos por todos los algoritmos en la red de complejidad alta. Se logra apreciar que los tiempos de ejecución son muy similares en todos los algoritmos con excepción de los algoritmos multi-objetivo NSGA-II y SPEA-2. Referente a las propuestas meméticas de los algoritmos NSGA-II y SPEA-2 se observa que, a diferencia de los anteriores experimentos, solo la propuesta que incluye al algoritmo HC como método de explotación logra un mejor valor de NRMS, aunque cabe resaltar que el tiempo de ejecución disminuye significativamente en comparación con los algoritmos multi-objetivo. Con respecto al algoritmo MOGBHS, se aprecia cómo, al igual que en las redes de complejidad baja y media, el mejor comportamiento se logra solo con el algoritmo multi-objetivo, sin la inclusión de un algoritmo de búsqueda local.

Algoritmo	NRMS				GEH del mejor resultado	Tiempo promedio (hh:mm)
	Mejor resultado	Peor resultado	Promedio	Desviación estándar		
HC	0,0887	0,1363	0,1080	0,014850969	93%	03:25
SA	0,1784	0,3318	0,2766	0,066797703	69%	03:24
ILS	0,0937	0,1277	0,1069	0,011912253	95%	03:35
NSGA-II	0,0804	0,1283	0,1029	0,015667031	93%	04:20
NSGA-II-HC	**0,0901**	**0,1195**	**0,1018**	**0,010196162**	**95%**	**03:32**
NSGA-II-SA	0,1065	0,1599	0,1293	0,019868558	92%	03:47
NSGA-II-ILS	0,0852	0,1513	0,1122	0,01856154	96%	03:46
MOGBHS	**0,0676**	**0,0883**	**0,0746**	**0,007524712**	**97%**	**03:22**
MOGBHS-HC	0,0739	0,1279	0,0872	0,015689967	97%	03:01

MOGBHS-SA	0,0809	0,1261	0,0969	0,014504441	95%	03:04
MOGBHS-ILS	0,0696	0,1182	0,0861	0,012578814	97%	03:13
SPEA-2	0,0795	0,1345	0,1050	0,01910512	96%	04:33
SPEA-2-HC	**0,0801**	**0,1092**	**0,0957**	**0,010866424**	**96%**	**03:38**
SPEA-2-SA	0,1167	0,1517	0,1348	0,012730534	95%	03:34
SPEA-2-ILS	0,0939	0,1191	0,1079	0,007194136	95%	03:51

Tabla 11: Resultados obtenidos por todos los algoritmos en la red I-75.

Para esta red todos los algoritmos, con excepción del algoritmo SA, cumplen con el criterio de calibración descrito en la sección 4.2. La anterior tabla muestra el estadístico GEH del mejor resultado encontrado en cada algoritmo, se logra apreciar que los algoritmos que cumplen con el criterio de calibración logran un valor de GEH menor a 5 para más del 90% de los enlaces. La Figura 51 ilustra los valores del estadístico GEH, antes y después del proceso de calibración de las propuestas que obtuvieron mejor promedio de NRMS de los algoritmos multi-objetivos. El estadístico GEH se realizó con los datos obtenidos por la ejecución que obtuvo el mejor resultado de cada una de las propuestas. La línea negra representa la condición inicial del modelo (antes de la calibración). El valor inicial de GEH era menor de 5 para aproximadamente el 40% de los enlaces. Las líneas roja, azul y verde representan la condición del modelo después de la calibración utilizando los algoritmos NSGA-II-HC, MOGBHS y SPEA-2-HC respectivamente.

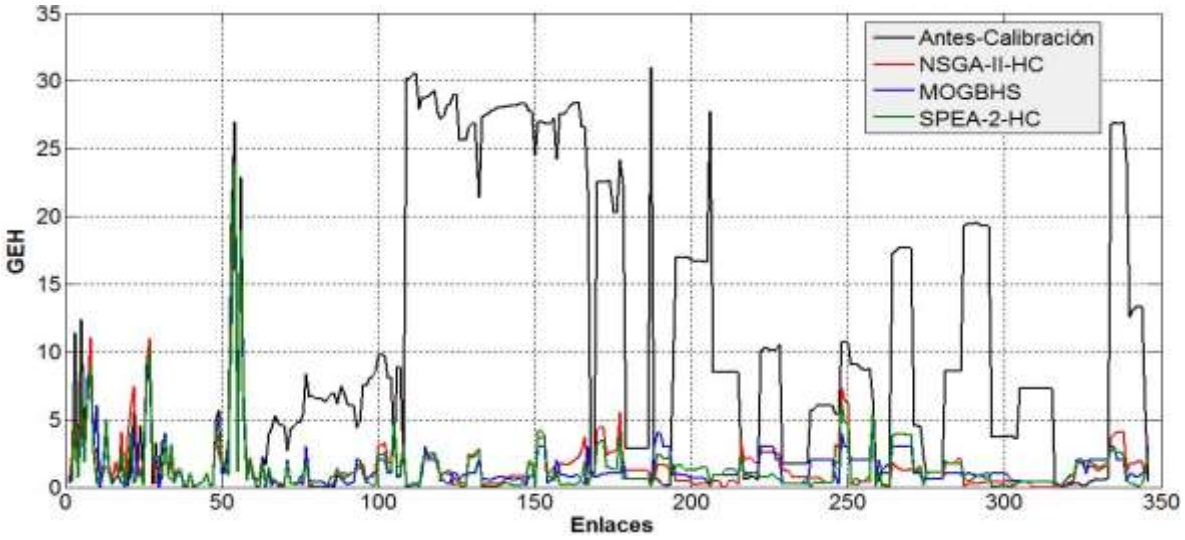

Figura 51: Estadística GEH en la red I-75 de los algoritmos NSGA-II-HC, MOGBHS y SPEA-2-HC.

5.4.3.4 COMPARACIÓN CON GASA Y SPSA

A continuación se presenta una comparación de la propuesta que obtiene mejores resultados en la red de complejidad alta con los algoritmos GASA y SPSA.

La Figura 52 muestra el promedio de las 30 ejecuciones de los algoritmos: MOGBHS, GASA y SPSA. Se logra apreciar cómo, similarmente a los anteriores experimentos, el algoritmo GASA obtiene mejores resultados al inicio de la ejecución, sin embargo el algoritmo MOGBHS, tiene un comportamiento más equilibrado, obtiene un mejor valor promedio de NRMS que los demás algoritmos y

su tiempo de ejecución es menor que GASA. Referente al algoritmo SPSA se observa como este algoritmo no logra obtener buenos resultados y este se estanca en valores iniciales de NRMS. El valor promedio de NRMS para GASA y SPSA es de 0,0961 y 0,3305 lo que equivale una mejora aproximada del 71% y escasamente 0,4% respectivamente.

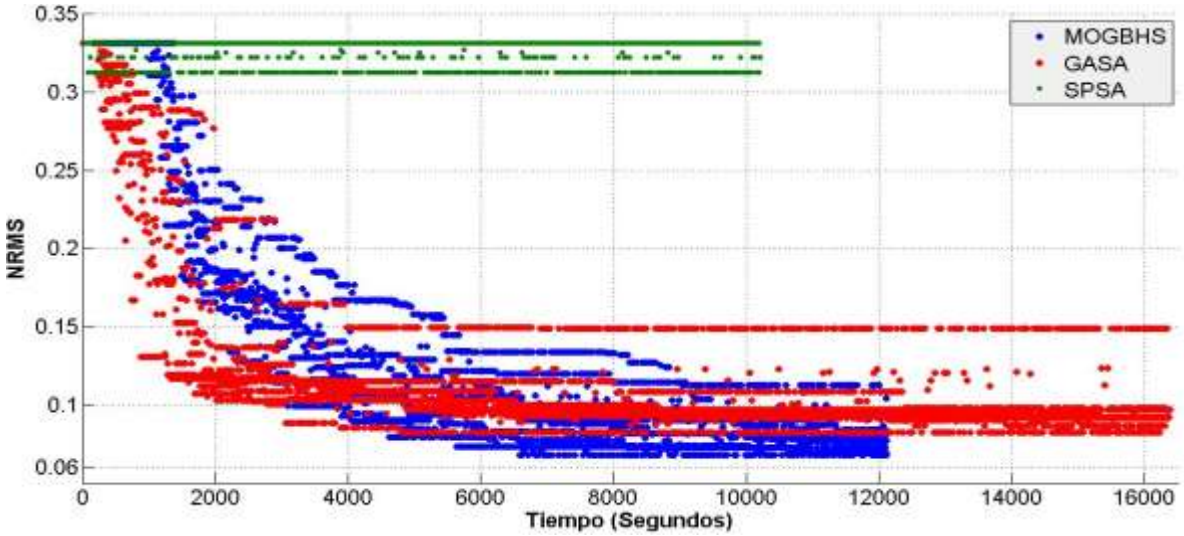

Figura 52: Gráfica NRMS vs Tiempo en la red I75 de la propuesta ganadora con los algoritmos del estado del arte.

Por último la Tabla 12 muestra un resumen de los resultados obtenidos por los algoritmos del estado del arte, GASA y SPSA en la red de complejidad alta. Con ella se logra analizar que el tiempo de ejecución de SPSA es menor que el de los algoritmos propuestos en esta investigación, sin embargo su promedio de ejecución es el más alto de todos, logrando mejoras insignificantes de NRMS y GEH. Referente a GASA, el tiempo de ejecución de este es mayor que todos los algoritmos propuestos excepto al multi-objetivo SPEA-2 con quien está a la par; el promedio de NRMS de GASA solo es superado por el algoritmo multi-objetivo MOGBHS y sus propuestas meméticas. El estadístico GEH del mejor resultado encontrado en cada algoritmo es mostrado en la Tabla 12; se logra apreciar como solo GASA cumple con el criterio de calibración, logrando un valor menor a 5 para el 93% de los enlaces.

Algoritmo	NRMS				GEH del mejor resultado	Tiempo promedio (hh:mm)
	Mejor resultado	Peor resultado	Promedio	Desviación estándar		
GASA	0,0821	0,1486	0,0961	0,015201182	93%	04:33
SPSA	0,3125	0,3318	0,3305	0,004964004	42%	03:00

Tabla 12: Resultados obtenidos por GASA y SPSA en la red I75.

5.4.3.5 ANÁLISIS ESTADÍSTICO

A continuación se presenta los resultados arrojados por las pruebas estadísticas de Wilcoxon y Friedman realizadas a todos los algoritmos propuestos y del estado del arte en la red de complejidad alta, con respecto a los valores de NRMS obtenidos al finalizar cada una de las 30 ejecuciones.

La Figura 53 muestra la dominancia de cada algoritmo con respecto a los demás. De esta figura se puede analizar que el algoritmo MOGBHS es el único que domina a todos los demás algoritmos. Por otro lado, similarmente a los anteriores experimentos, el algoritmo del estado del arte SPSA es dominado por todos los algoritmos. Así mismo el algoritmo GASA domina a los algoritmos SA, NSGA-II-SA, SPEA-2-SA y SPSA; y se destaca la dominancia sobre los algoritmos que tienen sus bases en SA, esto debido a que, como se pudo observar anteriormente el algoritmo SA propuesto en esta investigación no presenta un buen comportamiento en esta red. Finalmente el algoritmo GASA es dominado por MOGBHS y sus propuesta memética MOGBHS-ILS.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)
HC (1)	-	•				•		○	○	○	○			•		○	•
SA (2)	○	-	○	○	○	○	○	○	○	○	○	○	○	○	○	○	•
ILS (3)		•	-		○	•		○	○		○			•			•
NSGA-II (4)		•		-		•		○	○		○			•			•
NSGA-II-HC (5)		•	•		-	•		○	○		○			•	•		•
NSGA-II-SA (6)	○	•	○	○	○	-	○	○	○	○	○	○	○		○	○	•
NSGA-II-ILS (7)		•				•	-	○	○	○	○		○	•		○	•
MOGBHS (8)	•	•	•	•	•	•	•	-	•	•	•	•	•	•	•	•	•
MOGBHS-HC (9)	•	•	•			•	•	○	-	•		•		•	•		•
MOGBHS-SA (10)		•				•	•	○	○	-				•	•		•
MOGBHS-ILS (11)	•	•	•	•	•	•	•	○			-	•		•	•	•	•
SPEA-2 (12)		•						○	○			-		•			•
SPEA-2-HC (13)		•				•	•	○					-	•	•		•
SPEA-2-SA (14)	○	•	○	○	○		○	○	○	○	○	○	○	-	○	○	•
SPEA-2-ILS (15)		•				•		○	○	○	○		○	•	-		•
GASA (16)		•				•		○						•		-	•
SPSA (17)	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	-

Figura 53: Resultados de la prueba de Wilcoxon para la red I75.

La Tabla 13 muestra los resultados obtenidos por todos los algoritmos en la prueba de Friedman. Similarmente a los anteriores experimentos esta tabla ratifica que el algoritmo que presenta mejor comportamiento es el MOGBHS, de la misma forma se observa que el último lugar es ocupado por el algoritmo SPSA. Con respecto a los algoritmos meméticos de NSGA-II y SPEA-2 se puede observar que las mejores propuestas son las que incluyen al algoritmo HC como algoritmo de búsqueda local, mientras que las propuestas meméticas que incluyen a SA e ILS como algoritmos de búsqueda local no obtienen mejor Ranking que sus propuestas multi-objetivo. Por último se destaca como el algoritmo GASA logra mantener su ranking con respecto al anterior experimento.

ALGORITMO	RANKING
MOGBHS	1,6
MOGBHS-ILS	3,9
MOGBHS-HC	4
MOGBHS-SA	6,5
GASA	6,6
SPEA-2-HC	7,2
NSGA-II-HC	7,9
NSGA-II	8,1
SPEA-2	8,3

HC	9,4
ILS	9,5
SPEA-2-ILS	10,1
NSGA-II-ILS	10,4
NSGA-II-SA	12,8
SPEA-2-SA	13,7
SA	16,1
SPSA	16,9

Tabla 13: Resultados de la prueba de Friedman para la red I75.

5.5 ANÁLISIS DE RESULTADOS

Todos los algoritmos propuestos en esta investigación logran cumplir con el criterio de calibración en las redes de complejidad baja y media logrando un GEH, de su mejor resultado, menor a 5 para el 100% de los casos. En la red de complejidad alta solo el algoritmo de búsqueda local SA no logra cumplir con el criterio de calibración. Con respecto a los algoritmos del estado del arte, GASA cumple con el criterio de calibración en todos los experimentos mientras que SPSA no logra cumplir con dicho criterio en la red de complejidad alta.

Referente al promedio de NRMS y los test de Wilcoxon y Friedman, se destaca que el algoritmo MOGBHS fue superior en todos los experimentos. Las propuestas meméticas de NSGA-II y SPEA-2 que tuvieron mejor comportamiento fueron las que incluyen al algoritmo ILS como método de búsqueda local para las redes de complejidad baja y media y las que incluyen al algoritmo HC como método de búsqueda local para el modelo de complejidad alta. Se destaca como el algoritmo GASA obtiene un mejor comportamiento a medida que la complejidad de los experimentos aumenta; demostrando la eficiencia que tiene la estrategia usada por MOGBHS y GASA de tener un porcentaje para mover los puntos alrededor de la mejor solución; esto explica el por qué el algoritmo SA utilizado en esta investigación no logra buenos resultados para la red de complejidad alta, dado que este solo se mueve sobre la solución actual y no explota zonas aledañas a la mejor solución.

Los tiempos de ejecución de los algoritmos meméticos siempre fueron menores a los algoritmos multi-objetivos (manteniendo el número de EFO), además, es notorio como el método de búsqueda local ayuda a la rápida convergencia de los algoritmos meméticos, esto se hace más evidente a medida que la complejidad de los modelos aumenta.

Finalmente en la Figura 54 se muestran las poblaciones y frente de Pareto de las propuestas ganadoras de NSGA-II, MOGBHS, SPEA-2 para la red I75, dado que la complejidad de esta red la convierte en la más interesante para realizar un análisis de los frentes, cabe resaltar que dado el funcionamiento de SPEA-2 en este solo se muestra la población externa.

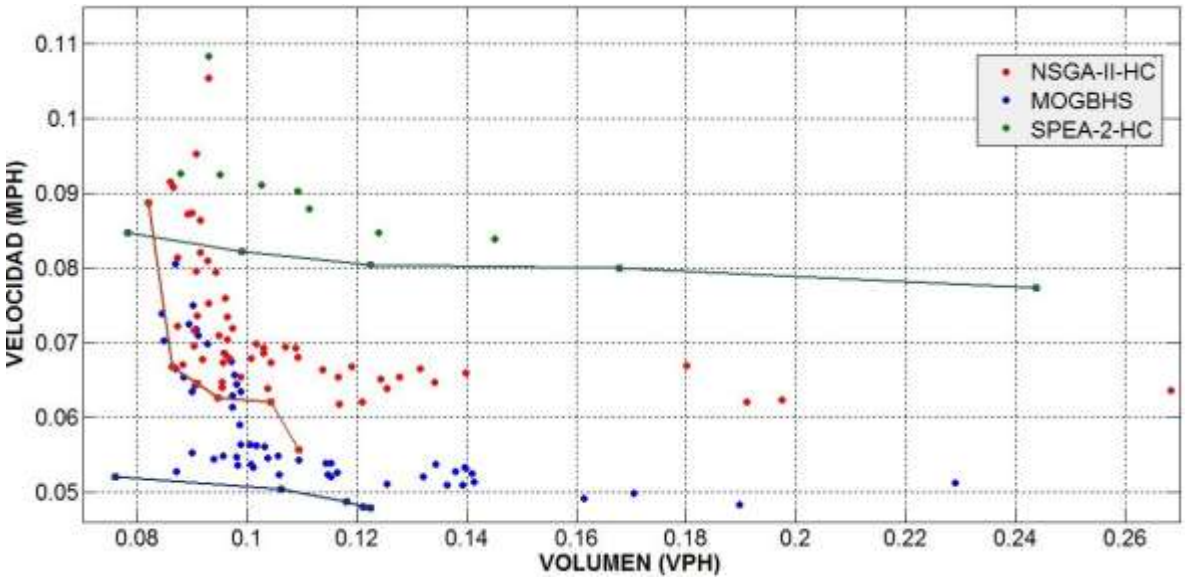

Figura 54: Frentes generados por los algoritmos NSGA-II-HC, MOGBHS y SPEA-2-HC en la red I-75.

Las líneas representan el frente de Pareto de cada algoritmo. Como se esperaba las soluciones del frente de Pareto se encuentran repartidas de manera uniforme, logrando una óptima exploración del espacio de búsqueda; se destaca como el algoritmo SPEA-2-HC es quien logra una mejor dispersión de los frentes, demostrando que la estrategia del algoritmo multi-objetivo SPEA-2 de utilizar la distancia al vecino más cercano es mejor que la distancia de apiñamiento utilizada por NSGA-II y MOGBHS para este problema en específico. Finalmente se destaca que en los tres casos la población restante tiende a estar cerca del frente de Pareto, lo que asegura que se está tratando con soluciones óptimas.

5.6 ANÁLISIS DE SENSIBILIDAD

El análisis de sensibilidad arroja información referente a qué parámetros son los que influyen más en la obtención de buenos resultados. En la presente investigación se realiza un análisis de sensibilidad al algoritmo que obtuvo mejores resultados en valor promedio de NRMS, el cual resultó ser el algoritmo MOGBHS. Para realizar este proceso se hizo uso del arreglo de cobertura descrito anteriormente, donde cada fila indica los parámetros de ejecución del algoritmo, con ellos se obtuvo el valor promedio de NRMS y un promedio del tiempo de ejecución. Con el objetivo de brindar una facilidad en la lectura de los resultados se categorizaron las anteriores variables de la siguiente forma:

Valor promedio de NRMS		**Tiempo promedio**	
Categoría	**Rango**	**Categoría**	**Rango (mm:ss)**
Excelente	NRMS <= 0,016	Excelente	Tiempo <= 14:00
Bueno	0,016 < NRMS <= 0,017	Regular	14:00 < Tiempo <= 17:00
Regular	NRMS > 0,017	Malo	Tiempo > 17:00

Tabla 14: Categorización de los valores promedio de NRMS y Tiempo.

Haciendo uso del algoritmo J-48 incluido en la herramienta Weka [85] y los resultados del CA, se obtiene el siguiente árbol de decisión:

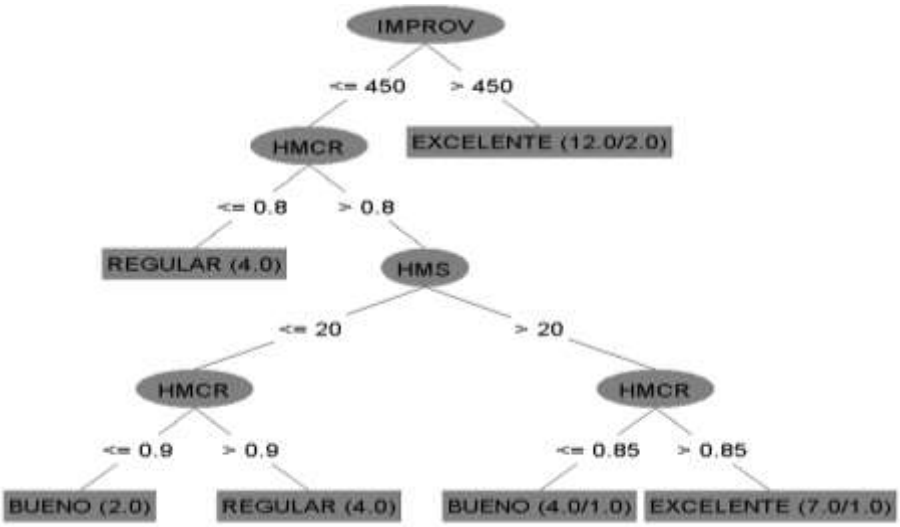

Figura 55: Árbol de decisión valor promedio de NRMS.

El primer análisis que se puede obtener del anterior árbol de decisión (Figura 55) es que si se cuenta con un número alto de improvisaciones (número de iteraciones) el algoritmo obtiene resultados de NRMS excelentes sin importar los valores de los demás parámetros. Cuando no se tiene un número de improvisaciones alto el algoritmo tiene en cuenta el valor del HMCR, si este es menor o igual a 0,8; es decir hay una probabilidad menor o igual del 80% de que se use la memoria armónica para generar una nueva armonía, el algoritmo obtiene resultados regulares de NRMS. Si el valor de HMCR supera el 0,8 se tiene en cuenta el tamaño de la memoria armónica, si este toma valores pequeños (menor o igual a 20) y valores menores o iguales a 0,9 de HMCR el algoritmo obtiene valores buenos de NRMS, mientras que si se utiliza un valor superior de 0,9 para HMCR el algoritmo obtiene valores regulares de NRMS. Esto se debe a que al trabajar con poblaciones pequeñas no se obtiene una buena exploración del espacio de soluciones. Por último si se trabaja con poblaciones relativamente altas (mayores de 20 individuos) el algoritmo tiende a resultados óptimos de NRMS estos son mejores a medida de que el valor de HMCR aumente.

De la Figura 55 se puede analizar que los valores que toman los parámetros Parmin y Parmax utilizados en el vocabulario del CA no influyen en la propuesta del algoritmo MOGBHS para el problema de calibración. Finalmente es válido recalcar que el anterior árbol de decisión no tiene en cuenta el tiempo promedio de ejecución del algoritmo. La Figura 56 ilustra el árbol de decisión obtenido tomando como clase objetivo el tiempo promedio de ejecución.

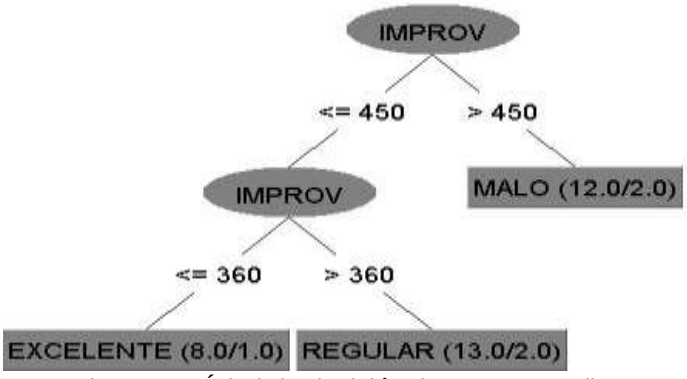

Figura 56: Árbol de decisión tiempo promedio.

De este árbol de decisión se puede analizar que el número de improvisaciones es el único parámetro influyente en el tiempo promedio de ejecución, obteniéndose que para improvisaciones mayores a 450, los tiempos serán malos, entre 360 y 450 serán regulares y para improvisaciones menores o iguales a 360, los tiempos serán excelentes.

De los dos árboles de decisión anteriores, se puede analizar que para obtener resultados óptimos tanto en valor de NRMS y tiempo de ejecución se debe contar con un tamaño de población superior a 20 individuos, manejar un valor de HMCR mayor a 0,85 y contar con un número de improvisaciones que no supere el valor de 360.

CAPÍTULO 5

6. CONCLUSIONES Y TRABAJOS FUTUROS

6.1 CONCLUSIONES

En este proyecto de investigación se adaptaron tres (3) algoritmos multi-objetivos (NSGA-II, MOGBHS y SPEA-2) a los cuales se le adaptaron, a cada uno, tres (3) algoritmos de búsqueda local (HC, SA e ILS) con el objetivo de convertir los algoritmos multi-objetivo en propuestas meméticas multi-objetivo para la calibración de modelos de flujo de tráfico vehicular CORSIM. Todas las propuestas fueron evaluadas sobre tres (3) modelos de diferentes complejidades. Los resultados arrojan que tanto los algoritmos multi-objetivo como los meméticos multi-objetivo cumplen con el criterio de calibración en todos los modelos. Adicionalmente los test de Friedman y Wilcoxon indican que el algoritmo que presenta mejor comportamiento estadístico es el MOGBHS.

Los resultados arrojados indican que la propuesta memética que obtiene mejores resultados de NRMS y en tiempos similares para los dos primeros modelos es la que incluye como algoritmo de explotación a la meta-heurística ILS mientras que en el modelo de complejidad alta la propuesta memética de NSGA-II y SPEA-2 es la que incluye como algoritmo de explotación al HC. Cabe resaltar que aunque el algoritmo HC original no cuenta con una estrategia para salir de óptimos locales, en las propuestas meméticas de esta investigación se optimiza desde varias soluciones, lo cual es un método recomendado en la literatura para evitar estancamientos. Por otro lado el mecanismo usado por la meta-heurística ILS es mejor para el problema de calibración de modelos de tráfico vehicular CORSIM en comparación con el utilizado por la meta-heurística multi-objetivo SA encontrada en el estado del arte [58].

Se realizó una comparación estadística de los algoritmos propuestos con los algoritmos del estado del arte GASA y SPSA (a los cuales se les adaptó el uso de hilos para lograr una comparación justa), demostrando que en todos los experimentos el algoritmo MOGBHS y sus propuestas meméticas son superiores estadísticamente a GASA mientras que el SPSA fue superado estadísticamente por todos los algoritmos propuestos en esta investigación. Por otro lado los resultados indicaron que la propuesta multi-objetivo de MOGBHS fue superior en todos los casos a las propuestas meméticas del mismo, este hecho se da porque, como se explicó anteriormente el MOGBHS implementa una estrategia similar a la del algoritmo de enjambre de partículas, la cual le permite realizar explotación alrededor de la mejor armonía (solución) encontrada (En este caso una armonía al azar perteneciente al frete de Pareto), por este motivo, la inclusión de los algoritmos de búsqueda local sobre el algoritmo MOGBHS, solo ayuda a la rápida convergencia

inicial del algoritmo, puesto que con estas propuestas se pierde una buena cantidad de explotaciones alrededor de las armonías del frente de Pareto dado que la cantidad de iteraciones de los algoritmos meméticos es menor que la del multi-objetivo y éste último tiende a realizar más explotación a medida que las iteraciones avanzan. Con esto, la propuesta ganadora MOGBHS fue comparada con respecto al tiempo de ejecución con los algoritmos del estado del arte, los resultados indican que el tiempo de ejecución de SPSA fue el menor en los tres experimentos seguido del algoritmo MOGBHS quien solo pierde en tiempo de ejecución con GASA en la red de complejidad baja. En referencia al tiempo de convergencia, GASA es quien obtiene mejores resultados al inicio de las ejecuciones, esto dado su estrategia de explotación, pero, aunque, MOGBHS demore un poco su convergencia a medida que la complejidad del experimento aumenta, este algoritmo es quien logra mejores resultados de NRMS y obtiene un comportamiento mucho más equilibrado.

Con lo anterior se logra cumplir con los objetivos planteados en la presente investigación.

Finalmente en esta investigación se realizó un afinamiento de parámetros y análisis de sensibilidad sobre el algoritmo MOGBHS (También se realizó un afinamiento de parámetros sobre el algoritmo NSGA-II-ILS), el cual fue el que obtuvo mejores resultados en todos los experimentos, los resultados obtenidos por el afinamiento de parámetros arrojaron que los parámetros utilizados para la ejecución del algoritmo MOGBHS fueron bastantes similares a los recomendados en la literatura, con excepción del tamaño de la memoria armónica. Con respecto al análisis de sensibilidad se demuestra que el algoritmo MOGBHS no es muy sensible a los cambios de los parámetros para este problema en específico.

6.2 TRABAJOS FUTUROS

En el presente proyecto de investigación se utilizaron tres (3) modelos de tráfico vehicular, con el objetivo de evaluar la eficacia de los algoritmos propuestos se hace necesario la inclusión de modelos que incluyen más enlaces y realizar experimentos con ellos, para así comprobar si los algoritmos propuestos logran la calibración sin importar la complejidad del modelo.

Realizar una modificación a las funciones objetivo de velocidad y volumen utilizando el acumulado de los valores absolutos en lugar de la raíz de los cuadrados, esto debido a que algunas propiedades del valor absoluto pueden ser más aplicables al problema de calibración de flujo de tráfico vehicular.

Adicionar un nuevo objetivo referente a la longitud de colas con el propósito de lograr una mejor representación de los modelos y comprobar que los algoritmos multi-objetivos propuestos logran calibrar modelos de tráfico vehicular con esta nueva información.

Por términos de tiempo, en esta investigación se realizó un afinamiento de parámetros únicamente a dos algoritmos de los quince (15) propuestos, los cuales fueron MOGBHS y NSGA-II-ILS. Si bien el afinamiento de ellos sirvió como base para las demás propuestas de estos algoritmos, para lograr mejores resultados se hace necesario realizar un afinamiento de parámetros a todos los algoritmos propuestos. Además de esto sería recomendable utilizar un vocabulario más variado dado que con ello se tiene un mayor espacio de búsqueda sobre los parámetros y utilizar un arreglo de cobertura de fuerza tres (3) o cuatro (4) con el fin de cubrir más casos de prueba.

Dado que el algoritmo MOGBHS se presenta como una versión multi-objetivo mejorada del algoritmo HS se pretende utilizar mejoras de los algoritmos NSGA-II y SPEA-2 presentes en el estado del arte, como los son los algoritmos ε-NSGA-II y SPEA-2+.

Finalmente el grupo de investigación espera como trabajo futuro desarrollar una comparación de diferentes algoritmos multi-objetivo para la calibración de modelos de flujo de tráfico vehicular, entre ellos: MOEA/D (Algoritmo evolutivo multi-objetivo basado en descomposición), OMOPSO (Algoritmo multi-objetivo basado en enjambre de partículas), entre otros.

CAPÍTULO 6

7. BIBLIOGRAFÍA

[1] F. H. Administration, "CORSIM User's Guide," 2006.

[2] S. Hoogendoorn and R. Hoogendoorn, "Calibration of microscopic traffic-flow models using multiple data sources," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences,* vol. 368, pp. 4497-4517, 2010.

[3] L. Yu, L. Yu, X. Chen, T. Wan, and J. Guo, "Calibration of VISSIM for bus rapid transit systems in Beijing using GPS data," *Journal of Public Transportation,* vol. 9, p. 13, 2006.

[4] R. Moussa and N. Chahinian, "Comparison of different multi-objective calibration criteria using a conceptual rainfall-runoff model of flood events," *Hydrology and Earth System Sciences,* vol. 13, pp. 519–535, 2009.

[5] P. Moscato and C. Cotta, "A modern introduction to memetic algorithms," in *Handbook of metaheuristics*, ed: Springer, 2010, pp. 141-183.

[6] A. Paz, V. Molano, E. Martinez, C. Gaviria, and C. Arteaga, "Calibration of traffic flow models using a memetic algorithm," *Transportation Research Part C: Emerging Technologies,* vol. 55, pp. 432-443, 2015.

[7] J. C. Spall, "An overview of the simultaneous perturbation method for efficient optimization," *Johns Hopkins apl technical digest,* vol. 19, pp. 482-492, 1998.

[8] J. J. Durillo and A. J. Nebro, "jMetal: A Java framework for multi-objective optimization," *Advances in Engineering Software,* vol. 42, pp. 760-771, 2011.

[9] D. Hadka, P. M. Reed, and T. W. Simpson, "Diagnostic assessment of the Borg MOEA for many-objective product family design problems," in *Evolutionary computation (CEC), 2012 IEEE congress on*, 2012, pp. 1-10.

[10] L. Ke, Q. Zhang, and R. Battiti, "MOEA/D-ACO: a multiobjective evolutionary algorithm using decomposition and antcolony," *Cybernetics, IEEE Transactions on,* vol. 43, pp. 1845-1859, 2013.

[11] M. Fadaee and M. Radzi, "Multi-objective optimization of a stand-alone hybrid renewable energy system by using evolutionary algorithms: A review," *Renewable and Sustainable Energy Reviews,* vol. 16, pp. 3364-3369, 2012.

[12] D. Kanagarajan, R. Karthikeyan, K. Palanikumar, and J. P. Davim, "Optimization of electrical discharge machining characteristics of WC/Co composites using non-dominated sorting genetic algorithm (NSGA-II)," *The International Journal of Advanced Manufacturing Technology,* vol. 36, pp. 1124-1132, 2008.

[13] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *Evolutionary Computation, IEEE Transactions on,* vol. 8, pp. 256-279, 2004.

[14] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *Evolutionary Computation, IEEE Transactions on,* vol. 6, pp. 182-197, 2002.

[15] C. A. P. Meneses and M. G. Echeverri, "Optimización multiobjetivo usando un algoritmo genético y un operador elitista basado en un ordenamiento no-dominado (NSGA-II)," *Scientia Et Technica,* vol. 1, 2007.

[16] N. V. Dung, B. Merz, A. Bárdossy, T. D. Thang, and H. Apel, "Multi-objective automatic calibration of hydrodynamic models utilizing inundation maps and gauge data," *Hydrology and Earth System Sciences,* vol. 15, pp. 1339-1354, 2011.

[17] S. Sivasubramani and K. Swarup, "Multi-objective harmony search algorithm for optimal power flow problem," *International Journal of Electrical Power & Energy Systems,* vol. 33, pp. 745-752, 2011.

[18] S. Salcedo-Sanz, D. Manjarres, Á. Pastor-Sánchez, J. Del Ser, J. A. Portilla-Figueras, and S. Gil-Lopez, "One-way urban traffic reconfiguration using a multi-objective harmony search approach," *Expert Systems with Applications,* vol. 40, pp. 3341-3350, 2013.

[19] V. Hajipour, S. H. A. Rahmati, S. H. R. Pasandideh, and S. T. A. Niaki, "A multi-objective harmony search algorithm to optimize multi-server location–allocation problem in congested systems," *Computers & Industrial Engineering,* vol. 72, pp. 187-197, 2014.

[20] F. Ruano, C. Cobos, and J. Torres-Jiménez, "Transit Network Frequencies-Setting Problem Solved Using a new Multi-Objective Global-Best Harmony Search Algorithm and Discrete Event Simulation," presented at the MICAI 2016 - 15th Mexican International Conference on Artificial Intelligence, Cancún, México, 2016.

[21] C. Cobos, C. Erazo, J. Luna, M. Mendoza, C. Gaviria, C. Arteaga, and A. Paz, "Multi-Objective Memetic Algorithm Based on NSGA-II and Simulated Annealing for Calibrating CORSIM Micro-Simulation Models of Vehicular Traffic Flow," *Lecture Notes in Artificial Intelligence,* 2016.

[22] A. Paz, V. Molano, and C. Gaviria, "Calibration of corsim models considering all model parameters simultaneously," in *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, 2012, pp. 1417-1422.

[23] J. Ma, H. Dong, and H. Zhang, "Calibration of microsimulation with heuristic optimization methods," *Transportation Research Record: Journal of the Transportation Research Board,* 2007.

[24] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," in *Eurogen*, 2001, pp. 95-100.

[25] K. P. Moses and M. D. Devadas, "An Approach to Reduce Root Mean Square Error in Toposheets," *European Journal of Scientific Research,* vol. 91, pp. 268-274, 2012.

[26] R. E. Shannon, "Introduction to the art and science of simulation," in *Proceedings of the 30th conference on Winter simulation*, 1998, pp. 7-14.

[27] S. G. Henderson and B. L. Nelson, "Stochastic computer simulation," *Handbooks in Operations Research and Management Science,* vol. 13, pp. 1-18, 2006.

[28] D. A. Wolf, "The role of microsimulation in longitudinal data analysis," *Canadian Studies in Population,* vol. 28, pp. 313-339, 2001.

[29] L. Bloomberg and J. Dale, "Comparison of VISSIM and CORSIM traffic simulation models on a congested network," *Transportation Research Record: Journal of the Transportation Research Board,* pp. 52-60, 2000.

[30] A. Kondyli, I. Soria, A. Duret, and L. Elefteriadou, "Sensitivity analysis of CORSIM with respect to the process of freeway flow breakdown at bottleneck locations," *Simulation Modelling Practice and Theory,* vol. 22, pp. 197-206, 2012.

[31] R. Moussa, N. Chahinian, and C. Bocquillon, "Distributed hydrological modelling of a Mediterranean mountainous catchment–Model construction and multi-site validation," *Journal of Hydrology,* vol. 337, pp. 35-51, 2007.

[32] Q. Duan, V. K. Gupta, and S. Sorooshian, "Shuffled complex evolution approach for effective and efficient global minimization," *Journal of optimization theory and applications,* vol. 76, pp. 501-521, 1993.

[33] H. Madsen, "Automatic calibration of a conceptual rainfall–runoff model using multiple objectives," *Journal of Hydrology,* vol. 235, pp. 276-288, 2000.

[34] P. Fabio, G. Aronica, and H. Apel, "Towards automatic calibration of 2-D flood propagation models," *Hydrology and Earth System Sciences,* vol. 14, pp. 911-924, 2010.

[35] E. Onieva, V. Milanés, J. Villagra, J. Pérez, and J. Godoy, "Genetic optimization of a vehicle fuzzy decision system for intersections," *Expert Systems with Applications,* vol. 39, pp. 13148-13157, 2012.

[36] T. Toledo, M. Ben-Akiva, D. Darda, M. Jha, and H. Koutsopoulos, "Calibration of microscopic traffic simulation models with aggregate data," *Transportation Research Record: Journal of the Transportation Research Board,* pp. 10-19, 2004.

[37] B. K. Abdalhaq and M. I. A. Baker, "Using meta heuristic algorithms to improve traffic simulation," *Journal of Algorithms,* vol. 2, pp. 110-128, 2014.

[38] A. J. Keane, "Genetic algorithm optimization of multi-peak problems: studies in convergence and robustness," *Artificial Intelligence in Engineering,* vol. 9, pp. 75-83, 1995.

[39] J. R. Schott, "Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization," DTIC Document1995.

[40] O. K. Erol and I. Eksin, "A new optimization method: big bang–big crunch," *Advances in Engineering Software,* vol. 37, pp. 106-111, 2006.

[41] P. Moscato and M. G. Norman, "A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems," *Parallel computing and transputer applications,* vol. 1, pp. 177-186, 1992.

[42] E. K. Burke, J. P. Newall, and R. F. Weare, "A memetic algorithm for university exam timetabling," in *International Conference on the Practice and Theory of Automated Timetabling*, 1995, pp. 241-250.

[43] C. C. Porras and P. Moscato, "Una introducción a los algoritmos meméticos," *Inteligencia artificial: Revista Iberoamericana de Inteligencia Artificial,* vol. 7, pp. 131-148, 2003.

[44] Z. Zhu, Y.-S. Ong, and M. Dash, "Wrapper–filter feature selection algorithm using a memetic framework," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics),* vol. 37, pp. 70-76, 2007.

[45] I. Kacem, S. Hammadi, and P. Borne, "Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic," *Mathematics and computers in simulation,* vol. 60, pp. 245-276, 2002.

[46] M. Huang, W. Han, J. Wan, Y. Ma, and X. Chen, "Multi-objective optimisation for design and operation of anaerobic digestion using GA-ANN and NSGA-II," *Journal of Chemical Technology and Biotechnology,* vol. 91, pp. 226-233, 2016.

[47] Z. W. Geem, J. H. Kim, and G. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation,* vol. 76, pp. 60-68, 2001.

[48] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Applied mathematics and computation,* vol. 188, pp. 1567-1579, 2007.

[49] M. G. Omran and M. Mahdavi, "Global-best harmony search," *Applied mathematics and computation,* vol. 198, pp. 643-656, 2008.

[50] Q.-K. Pan, P. N. Suganthan, M. F. Tasgetiren, and J. J. Liang, "A self-adaptive global best harmony search algorithm for continuous optimization problems," *Applied Mathematics and Computation,* vol. 216, pp. 830-848, 2010.

[51] F. Ruano, "Transit Network Frequencies-Setting Problem Solved Using a new Multi-Objective Global-Best Harmony Search Algorithm and Discrete Event Simulation," Master Thesis, Computer Science, Universidad del Cauca, 2016.

[52] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary computation,* vol. 8, pp. 173-195, 2000.

[53] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," *IEEE transactions on evolutionary computation,* vol. 7, pp. 204-223, 2003.

[54] F. Neri, C. Cotta, and P. Moscato, *Handbook of memetic algorithms* vol. 379: Springer, 2012.

[55] S. Sakamoto, E. Kulla, T. Oda, M. Ikeda, L. Barolli, and F. Xhafa, "A comparison study of Hill Climbing, Simulated Annealing and Genetic Algorithm for node placement problem in WMNs," *Journal of High Speed Networks,* vol. 20, pp. 55-66, 2014.

[56] M. Nandhini and S. Kanmani, "A survey of simulated annealing methodology for university course timetabling," *International Journal of Recent Trends in Engineering,* vol. 1, pp. 255-257, 2009.

[57] R. Eglese, "Simulated annealing: a tool for operational research," *European Journal of Operational Research,* vol. 46, pp. 271-281, 1990.

[58] D. Nam and C. H. Park, "Multiobjective simulated annealing: A comparative study to evolutionary algorithms," *International Journal of Fuzzy Systems,* vol. 2, pp. 87-97, 2000.

[59] S. Luke, *Essentials of metaheuristics*: Lulu Com, 2013.

[60] J. Torres-Jimenez, I. Izquierdo-Marquez, R. N. Kacker, and D. R. Kuhn, "Tower of covering arrays," *Discrete Applied Mathematics,* vol. 190, pp. 141-146, 2015.

[61] E. Brockfeld, R. Kühne, and P. Wagner, "Calibration and validation of microscopic traffic flow models," *Transportation Research Record: Journal of the Transportation Research Board,* pp. 62-70, 2004.

[62] T. Y. Gan and G. F. Biftu, "Automatic calibration of conceptual rainfall-runoff models: Optimization algorithms, catchment conditions, and model structure," *Water resources research,* vol. 32, pp. 3513-3524, 1996.

[63] V. Punzo and B. Ciuffo, "How parameters of microscopic traffic flow models relate to traffic dynamics in simulation: Implications for model calibration," *Transportation Research Record: Journal of the Transportation Research Board,* pp. 249-256, 2009.

[64] Y. Gardes, A. D. May, J. Dahlgren, and A. Skabardonis, "Freeway calibration and application of the PARAMICS model," in *81st Annual Meeting of the Transportation Research Board, Washington, DC*, 2002.

[65] B. Park and J. Schneeberger, "Microscopic Simulation Model Calibration and validation: case study of VISSIM simulation model for a coordinated actuated Signal system," *Transportation Research Record: Journal of the Transportation Research Board,* pp. 185-192, 2003.

[66] K.-O. Kim and L. Rilett, "Simplex-based calibration of traffic microsimulation models with intelligent transportation systems data," *Transportation Research Record: Journal of the Transportation Research Board,* pp. 80-89, 2003.

[67] T. Ma and B. Abdulhai, "Genetic algorithm-based optimization approach and generic tool for calibrating traffic microscopic simulation parameters," *Transportation Research Record: Journal of the Transportation Research Board,* pp. 6-15, 2002.

[68] A. L. Cunha, J. E. Bessa Jr, and J. R. Setti, "Genetic algorithm for the calibration of vehicle performance models of microscopic traffic simulators," in *Progress in Artificial Intelligence*, ed: Springer, 2009, pp. 3-14.

[69] B. Park and H. Qi, "Development and Evaluation of a Procedure for the Calibration of Simulation Models," *Transportation Research Record: Journal of the Transportation Research Board,* pp. 208-217, 2005.

[70] V. Vaze, C. Antoniou, Y. Wen, and M. Ben-Akiva, "Calibration of dynamic traffic assignment models with point-to-point traffic surveillance," *Transportation Research Record: Journal of the Transportation Research Board,* pp. 1-9, 2009.

[71] M. Kontorinaki, A. Spiliopoulou, I. Papamichail, M. Papageorgiou, Y. Tyrinopoulos, and J. Chrysoulakis, "Overview of nonlinear programming methods suitable for calibration of traffic flow models," *Operational Research,* vol. 15, pp. 327-336, 2015.

[72] B. Park and J. Lee, "Optimization of coordinated-actuated traffic signal system: Stochastic optimization method based on shuffled frog-leaping algorithm," *Transportation Research Record: Journal of the Transportation Research Board,* pp. 76-85, 2009.

[73] R. Putha, L. Quadrifoglio, and E. Zechman, "Comparing ant colony optimization and genetic algorithm approaches for solving traffic signal coordination under oversaturation conditions," *Computer-Aided Civil and Infrastructure Engineering,* vol. 27, pp. 14-28, 2012.

[74] B. Ciuffo, V. Punzo, and V. Torrieri, "Comparison of simulation-based and model-based calibrations of traffic-flow microsimulation models," *Transportation Research Record: Journal of the Transportation Research Board,* pp. 36-44, 2008.

[75] C. Cobos, C. Daza, C. Martínez, M. Mendoza, C. Gaviria, C. Arteaga, and A. Paz, "Calibration of Microscopic Traffic Flow Simulation Models Using a Memetic Algorithm with Solis and Wets Local Search Chaining (MA-SW-Chains)," in *Ibero-American Conference on Artificial Intelligence*, 2016, pp. 365-375.

[76] M. Shafii and F. D. Smedt, "Multi-objective calibration of a distributed hydrological model (WetSpa) using a genetic algorithm," *Hydrology and Earth System Sciences,* vol. 13, pp. 2137-2149, 2009.

[77] Y. Tang, P. Reed, and T. Wagener, "How effective and efficient are multiobjective evolutionary algorithms at hydrologic model calibration?," *Hydrology and Earth System Sciences Discussions,* vol. 10, pp. 289-307, 2006.

[78] J. Bonilla, L. J. Yebra, S. Dormido, and E. Zarza, "Parabolic-trough solar thermal power plant simulation scheme, multi-objective genetic algorithm calibration and validation," *Solar Energy,* vol. 86, pp. 531-540, 2012.

[79] K. Behzadian, Z. Kapelan, D. Savic, and A. Ardeshir, "Stochastic sampling design using a multi-objective genetic algorithm and adaptive neural networks," *Environmental Modelling & Software,* vol. 24, pp. 530-541, 2009.

[80] X. Wang, C. Hirsch, S. Kang, and C. Lacor, "Multi-objective optimization of turbomachinery using improved NSGA-II and approximation model," *Computer Methods in Applied Mechanics and Engineering,* vol. 200, pp. 883-895, 2011.

[81] Y.-K. Lin and C.-T. Yeh, "Multi-objective optimization for stochastic computer networks using NSGA-II and TOPSIS," *European Journal of Operational Research,* vol. 218, pp. 735-746, 2012.

[82] P. Holm, D. Tomich, J. Sloboden, and C. Lowrance, "Traffic analysis toolbox volume iv: guidelines for applying corsim microsimulation modeling software," 2007.

[83] D. Hadka, "MOEA Framework: a Free and Open Source Java Framework for Multiobjective Optimization. User Manual. Version 2.4," ed. http://www.moeaframework.org/. 2014.

[84] J. Alcalá, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *Journal of Multiple-Valued Logic and Soft Computing,* vol. 17, pp. 255-287, 2010.

[85] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *ACM SIGKDD explorations newsletter,* vol. 11, pp. 10-18, 2009.