UNIVERSIDAD DE DEUSTO

APLICACIÓN DE TÉCNICAS DE INTELIGENCIA ARTIFICIAL PARA LA PREDICCIÓN DE CONGESTIONES A CORTO PLAZO

Tesis doctoral presentada por Pedro López García

dentro del Programa de Doctorado en Ing. para la Sociedad de la Información y Desarrollo Sostenible

Dirigida por Dr. Enrique Onieva Caracuel y
Dr. Asier Perallos Ruiz

UNIVERSIDAD DE DEUSTO

APLICACIÓN DE TÉCNICAS DE INTELIGENCIA ARTIFICIAL PARA LA PREDICCIÓN DE CONGESTIONES A CORTO PLAZO

Tesis doctoral presentada por Pedro López García

dentro del Programa de Doctorado en Ing. para la Sociedad de la Información y Desarrollo Sostenible

Dirigida por Dr. Enrique Onieva Caracuel y
Dr. Asier Perallos Ruiz

El doctorando | El director | El director

Bilbao, Junio de 2016

Aplicación de técnicas de Inteligencia Artificial para la predicción de congestiones a corto plazo

Autor: Pedro López García
Directores: Enrique Onieva y Asier Perallos

Texto impreso en Bilbao
Primera edición, Junio de 2016

Índice general

Índice de figuras

Índice de tablas

ACO Optimización de colonia de hormigas, del inglés *Ant Colony Optimization*

ARIMA Autoregressive Integrated Moving Average

CE Entropía Cruzada, del inglés *Cross Entropy*

CMA Adaptación de la Matriz de Covarianza, del inglés *Covariance Matrix Adaptation*

DE Evolución Diferencial, del inglés *Differential Evolution*

FRBS Sistemas basados en Reglas Difusas, del inglés *Fuzzy Rule-Based Systems*

GA Algoritmos Genéticos, del inglés *Genetic Algorithms*

HFRBS Sistemas Basados en Reglas Difusas Jerárquicos, del inglés *Hierarchical Fuzzy Rule-Based Systems*

ITS Sistemas Inteligentes de Transporte, del inglés *Intelligent Transportation Systems*

KF Filtro de Kalman, del inglés *Kalman Filter*

NN Redes Neuronales, del inglés *Neural Networks*

PSO Algoritmo de Optimización de Partículas, del inglés *Particle Swarm Optimization*

V2V Vehículo a Vehículo

SA Recocido Simulado, del inglés *Simulated Annealing*

SVM Máquinas Vector Soporte, del inglés *Support Vector Machines*

TOD Time-of-Day

TSP Traveling Salesman Problem

La gente prefiere olvidar lo imposible;
les hace la vida más fácil.

Neil Gaiman

1

Introducción

El transporte y la movilidad juegan un papel fundamental en nuestra vida diaria. Acciones como viajar, el transporte de mercancías o pasajeros, o incluso ir al trabajo forman parte de nuestro día a día. La mayoría de los desplazamientos que realizamos se realizan por carretera, como se puede apreciar de los datos aportados por el último eurobarómetro sobre la calidad del transporte emitido por la Comisión Europea [Commission 14]. De este estudio, podemos extraer como primera conclusión que el método más utilizado para la mayoría de nuestros desplazamientos diarios es, sin duda alguna, el coche particular. La Figura 1.1 muestra la gran diferencia existente entre el uso de este medio de transporte en comparación con el resto en lo que al día a día se refiere ya que más de la mitad de los desplazamientos se hacen en este tipo de transporte. A estos datos hay que añadir que el coche es, por encima de medios de transporte como el avión o el tren, uno de los más utilizados en los desplazamientos de larga distancia (al menos, 300 km). Centrándonos en el transporte por carretera, en términos de calidad, un 38 % de los encuestados opina que ha mejorado en los últimos 5 años en su país. A la par, un 36 % opina todo lo contrario.

A la pregunta de '¿cuál de los siguientes aspectos piensa que es uno de los problemas más serios que afectan a las carreteras?', y siendo posible un máximo de

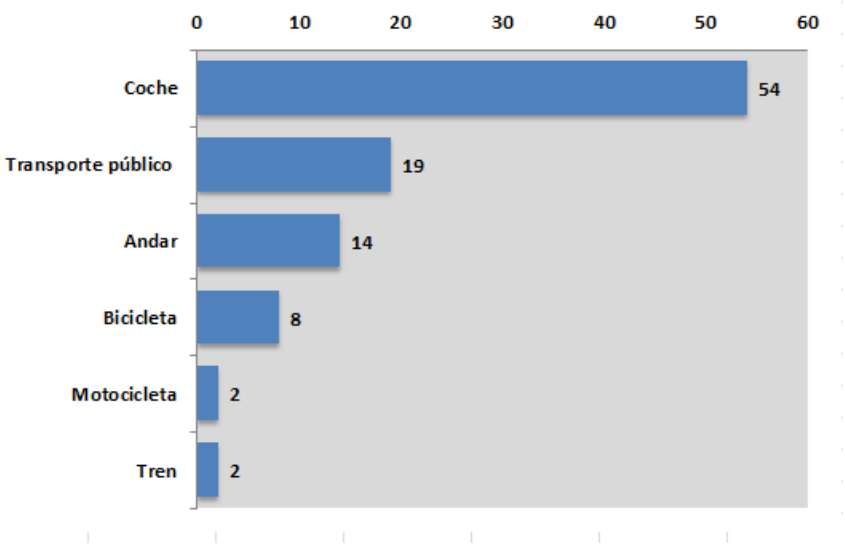

Figura 1.1: Valores en porcentaje (eje X) de las respuestas a la pregunta *En un día normal, ¿qué método de transporte utiliza normalmente?* realizada en el Eurobarómetro de 2014 sobre Transporte.

tres respuestas, un 60 % de los encuestados eligió la congestión de tráfico como una de sus opciones, seguido por un 59 % que eligió el mantenimiento de la carretera. El resto de respuestas y sus porcentajes se muestran en la Figura 1.2.

Los Sistemas Inteligentes de Transporte, del inglés *Intelligent Transportation Systems* (ITS) se definen como la intersección de las tecnologías de la comunicación y la electrónica con el fin de mitigar los problemas del transporte, ya sea terrestre, marítimo o aéreo. Tecnologías como las comunicaciones vehiculares, los sistemas de control de señales de tráfico, cámaras que detectan la velocidad o la matrícula automáticamente, o los sistemas de información en carretera son algunas de las áreas de aplicación de los ITS.

La congestión del tráfico y su temprana detección es otro tema fundamental en este campo de investigación. Uno de los objetivos de los ITS en este ámbito es realizar una predicción satisfactoria que pueda ayudar a tomar medidas para, por ejemplo, la disminución de ruido, no sólo en entornos urbanos, sino también en autopistas, el ahorro de energía, resultado del descenso de la polución, el incremento de la eficacia y el rendimiento de los sistemas de transporte, y ahorro en infraestructura pública para las instituciones.

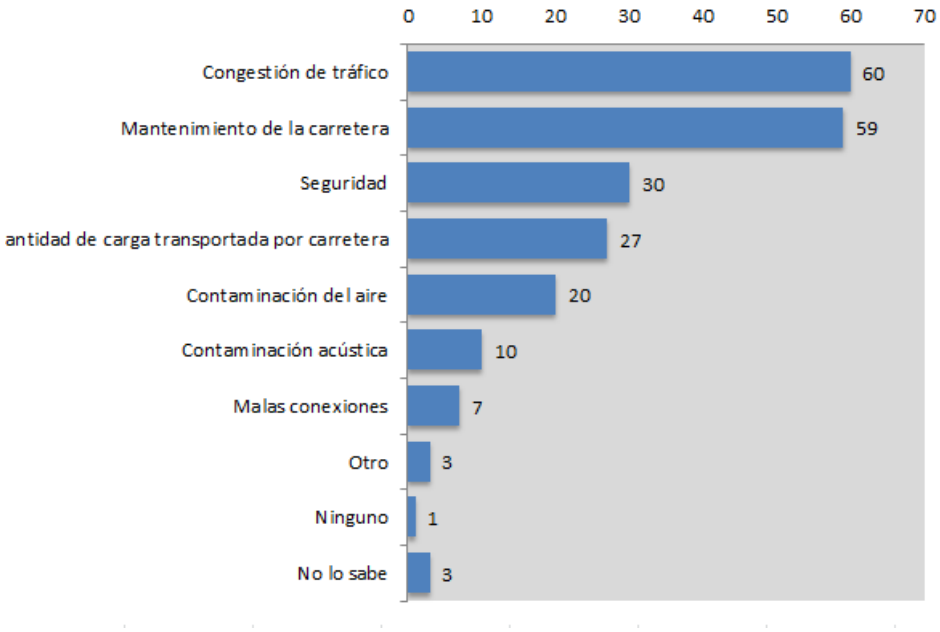

Figura 1.2: Valores en porcentaje (eje X) de las respuestas a la pregunta *¿Cuál de los siguientes aspectos piensa que es uno de los problemas más serios que afectan a las carreteras?* realizada en el Eurobarómetro de 2014 sobre Transporte.

Para realizar predicciones del estado futuro del tráfico, dos de los métodos más usados en la previsión de tráfico en la última década han sido el Filtro de Kalman, del inglés *Kalman Filter* (KF) [Zhang 12b] y Autoregressive Integrated Moving Average (ARIMA) [Tan 07]. El KF utiliza un conjunto de ecuaciones que proporcionan un medio de cálculo eficiente para estimar el estado del proceso de forma que minimice el error medio [Welch 95]. El filtro tiene sus ventajas en varios aspectos: mantiene estimaciones pasadas, presentes e incluso de futuros estados y puede hacerlo incluso cuando el modelo del sistema es desconocido. Por otro lado, ARIMA es una generalización del modelo ARMA (AutoRegressive Moving Average) [Box 13], adaptado para series temporales de manera que pueda predecir puntos futuros en dichas series.

Ambos son modelos regresivos que encuentran patrones en los datos pasados para la predicción de un valor en el futuro. Su uso no sólo se reduce a los ITS [Williams 98, Ahmed 79, Yang 07, Okutani 84], sino que se ha extendido a otros campos. Por ejemplo, ARIMA se ha usado en la predición de la calidad del agua

en [Ömer Faruk 10], en la previsión de la demanda eléctrica en [Taylor 06] y en la demanda de turismo en [Chen 12]. En el caso del KF, el método ha sido usado en [De Ridder 13] para determinar la calidad del aire y en [Saha 10] para la previsión del clima. Además, estos dos métodos pueden ser combinados, como se puede ver en [Liu 12]. Sin embargo, estos procesos tienen varios problemas. Para los KFs tradicionales, la precisión de la predicción puede ser inestable [Wang 05]. ARIMA puede ser incapaz de predecir correctamente series temporales de flujo de tráfico dado que están basados en teoría de procesos estocásticos estacionarios en los que normalmente se realizan supuestos [Tan 07].

En los últimos años, se han desarrollado otras alternativas. En [Bauza 13], la detección de la congestión se realizó a través del uso de comunicaciones Vehículo a Vehículo (V2V) en autopistas de largo recorrido. Otro ejemplo de previsión de congestión usando un esquema de comunicaciones vehiculares se presentó en [Xu 10], donde se propone el uso de varias redes de transmisión. En [Rzeszotko 12], diferentes técnicas de minería de datos son usadas para predecir la velocidad media en una calle y ruta dada por vehículos individuales para evitar retenciones de tráfico. Otro tipo de esquema es el llamado Time-of-Day (TOD). Un ejemplo es utilizado en [Jia 06], donde el control TOD divide un día en varios intervalos e intenta encontrar los controladores óptimos para cada intervalo.

Otra clase de alternativas son las técnicas de Soft Computing como Máquinas Vector Soporte, del inglés *Support Vector Machines* (SVM), Redes Neuronales, del inglés *Neural Networks* (NN), Algoritmos Genéticos, del inglés *Genetic Algorithms* (GA) o Sistemas basados en Reglas Difusas, del inglés *Fuzzy Rule-Based Systems* (FRBS). Estas técnicas se han usado por separado, como en [Castro-Neto 09], donde se utiliza una SVM de regresión online para predecir la fluidez del tráfico a corto plazo teniendo en cuenta las condiciones del medio, o en [Li 11] donde, entre otros modelos, se utiliza una NN para predecir el tiempo de viaje de motoristas; o combinando diferentes métodos como es el caso de [Stathopoulos 08], donde se utiliza un FRBS con una metaheurística para optimizar los parámetros de los sistemas y predecir el tráfico a corto plazo, de [Gu 12], donde se combinan un GA y una NN Difusa para predicción de tráfico, de [Gu 11], donde una SVM se combina con un método de descomposición empírico y se optimiza con un Algoritmo de Optimización de Partículas, del inglés *Particle Swarm*

Optimization (PSO) para la predicción de tráfico en tiempo real, o de [Zhang 14], donde se combinan FRBS con GAs para la predicción de congestión a corto plazo.

Siguiendo con estas técnicas, SVMs se encuentran con dificultades para obtener valores precisos cuando tratan con una gran cantidad de variables. Por otro lado, las NN obtienen buenos resultados y han atraído más atención a lo largo de los años. Sin embargo, debido al problema de óptimo local y su poca habilidad de generalización, su eficacia es, en algunos casos, limitada. Además, el resultado de una NN depende no sólo del entrenamiento de la red sino de la cantidad de datos de calidad disponibles y de los parámetros definidos.

Por otra parte, la aplicación de métodos probabilísticos como el método de la Entropía Cruzada, del inglés *Cross Entropy* (CE) [Rubinstein 99] o el método de Adaptación de la Matriz de Covarianza, del inglés *Covariance Matrix Adaptation* (CMA) [Hansen 01] en este tipo de problemas se contempla en artículos como [Lopez-Garcia 16a], donde CE se utiliza junto a otros métodos para la predicción de la congestión en una autopista, o en [Stellet 15], donde CMA se aplica a sistemas de asistencia al conductor.

El trabajo que nos ocupa tiene como principal objetivo la utilización de técnicas de Soft Computing, concrétamente FRBS y GA, para la previsión de congestiones de tráfico a corto plazo en autopistas y autovías. Además, se utilizan métodos probabilísticos, como es el caso de CE nombrado anteriormente, para la mejora del rendimiento del resto de técnicas usando su capacidad de explotación. Así pues, se ha desarrollado un algoritmo híbrido que combina GA y CE, llamado GACE por sus siglas en inglés, siendo este la principal aportación de esta tesis.

Todas las técnicas de las que se ha hablado durante esta introducción son utilizadas en muchos y variados ámbitos. Uno de ellos, la optimización, ya sea de problemas del mundo real [Lu 14, DellÁmico 15] como creados artificialmente [Syberfeldt 09, Wang 09] es un tema que siempre ha sido bien recibido y genera una creciente atención en la comunidad científica. El reto que supone resolver los problemas antes especificados es una motivación para su estudio, sobre todo en el área de la inteligencia artificial. Hay muchos tipos de optimización, entre los que destacan la optimización numérica [Schwefel 81], lineal [Bertsimas 97], continua [Eiselt 87] o combinatoria [Papadimitriou 98]. Este tema se ampliará en secciones

posteriores, donde aplicaremos el algoritmo desarrollado a esta clase de problemas para demostrar su utilidad en otros campos.

1.1 Propósito e hipótesis

Poder prever la formación de congestiones en autopistas y autovías a corto plazo es un reto para los ITS. Realizar dicha predicción correctamente conllevaría, como se ha hablado en el apartado anterior, la disminución de la polución, el ruido y el tiempo de viaje así como un aumento de la productividad en el transporte de mercancías y personas.

El propósito de la tesis presentada en esta memoria está motivado por la intención de predecir las posibles retenciones a corto plazo que se den en todo tipo de carreteras. Una de las ventajas de realizar dicha predicción a corto plazo es la adaptación de la toma de decisiones al momento exacto en el que ocurren los sucesos. Por ejemplo, si se produce un accidente en la carretera por la que circulamos, es posible que se produzca una retención al poco tiempo. En el caso de realizar esta predicción a largo plazo, no se podrían prever las posibles situaciones inesperadas en las carreteras. Otra de las ventajas es la capacidad de organizar un desplazamiento antes de comenzarlo. Si el usuario conoce si existirán o no retenciones en la vía que forma parte de su ruta, podrá organizar, incluso durante el desplazamiento, la ruta que le lleve a utilizar el menor tiempo posible para llegar a su destino. Esto, aunque también se puede llevar a cabo con una predicción a largo plazo, lleva a que el usuario no se tenga que preocupar por su ruta hasta el momento anterior de comenzarla. Para llevar a cabo dicho propósito, no sólo se ha trabajado en la predicción de la congestión en un único punto sino a lo largo de toda la sección de una autopista, dando un nuevo enfoque a este tipo de problema. Para alcanzar este objetivo se propone el uso de Sistemas Basados en Reglas Difusas Jerárquicos, del inglés *Hierarchical Fuzzy Rule-Based Systems* (HFRBS). Para optimizar los parámetros de cada sistema de la jerarquía, se combina un algoritmo de CE con un GA, proporcionando así tanto una mejora tanto en la exploración del espacio de búsqueda como la explotación de los individuos de la población.

A su vez, y para demostrar que el algoritmo propuesto para la optimización de los sistemas puede también aplicarse a otros ámbitos de la literatura, se utiliza

dicho algoritmo para optimizar diferentes funciones matemáticas de complejidad creciente.

Dados estos propósitos, la hipótesis que se quiere demostrar con la realización de este trabajo es la siguiente:

La optimización de jerarquías de Sistemas Basados en Reglas Difusas así como de los sistemas que las forman a partir de la hibridación de Algoritmos Evolutivos con Métodos Probabilísticos ayudará a predecir la formación de congestiones de tráfico a corto plazo utilizando diferentes tipos de punto de interés. A su vez, esta hibridación podrá adaptarse a otros ámbitos, como puede ser la optimización de funciones, de cara a obtener un rendimiento aceptable en éstos.

1.2 Motivación

La motivación sobre la que se ha edificado esta tesis, sin entrar en detalle sobre las partes utilizadas, es la de crear un algoritmo híbrido que sea capaz, no sólo de ayudar a los sistemas inteligentes a predecir de manera lo más fiable posible las congestiones de tráfico que se puedan producir en entornos urbanos o carreteras, sino que se pueda aplicar a otro tipo de problemas diferentes, como pueden ser problemas de optimización, o elección de clasificadores de un ensemble, previo paso del estudio del problema en cuestión y adaptación de sus características a dicho problema.

1.3 Difusión de los resultados

Durante los años en los que se ha realizado el proyecto que conlleva como colofón final la escritura de este documento, los resultados que se han ido obteniendo con las diferentes experimentaciones y problemas se han ido difundiendo en diferentes congresos, a nivel nacional e internacional, y revistas con alto índice de impacto. A continuación, se listan las publicaciones derivadas del trabajo de tesis aquí presentado:

- **P. Lopez-Garcia**, E. Onieva, A. Perallos, *Optimización de Sistemas Basados en Reglas Difusas para la predicción de congestión a corto plazo*, Jornadas científico-técnicas y seminario doctoral SEMATICA 2014.

- ◇ **P. Lopez-Garcia**, E. Onieva, A. Perallos, L. Arjona, E. Osaba, *Optimización de Sistemas Basados en Reglas Difusas para la predicción de congestión a corto plazo*, Congreso Nacional sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB), 2015, en los Proceedings, páginas 621–628.

- ◇ **P. Lopez-Garcia**, E. Onieva, E. Osaba, A. Masegosa, A. Perallos, *Hybridizing Genetic Algorithm with Cross Entropy for Solving Continuous Functions*, Genetic and Evolutionary Computation Conference, 2015, en los Proceedings, páginas 763–764.

- ◇ **P. Lopez-Garcia**, E. Onieva, E. Osaba, A. Masegosa, A. Perallos, *A hybrid method for short-term traffic congestion forecasting using genetic algorithms and cross-entropy*, IEEE Transactions Intelligent Transportation Systems, 2016, Volumen 17, Issue 2, Páginas 557-569. Esta revista cuenta con un factor de impacto de 2.377 en el Journal Citations Report del año 2014, y se encuentra indexada en las siguientes posiciones:

  1. Posición 13 de 125 de la categoría Civil Engineering, cuartil Q1.
  2. Posición 41 de 249 en la categoría Electrical & Electronic Engineering, cuartil Q1.
  3. Posición 9 de 33 de la categoría Transportation Science & Technology, cuartil Q2.

- ◇ **P. Lopez-Garcia**, E. Onieva, E. Osaba, A.D. Masegosa, A. Perallos, *GACE: A meta-heuristic based in the hybridization of Genetic Algorithms and Cross Entropy methods for continuous optimization*, Expert Systems with Applications. 2016, Volumen 55, páginas 508–519. Esta revista cuenta con un factor de impacto de 2.240 en el Journal Citations Report del año 2014, así como ocupa las siguientes posiciones en diferentes categorías:

  1. Posición 29 de 123 de la categoría Computer Science, Artificial Intelligence, cuartil Q1.
  2. Posición 48 de 249 de la categoría Electrical & Electronic Engineering, cuartil Q1.

3. Posición 12 de 81 de la categoría Operations Research & Management Science, cuartil Q1.

También, y para quien sea de interés, se añade una lista con los diferentes artículos en los que ha participado el doctorando como co-autor a lo largo del trabajo realizado:

- ◇ A. D. Masegosa, A. Bahillo, E. Onieva, **P. Lopez-Garcia**, A. Perallos, *A new optimization approach for indoor location based on Differential Evolution*, International Fuzzy Systems Association (IFSA) 2015.

- ◇ E. Osaba, F. Diaz, E. Onieva, **P. Lopez-Garcia**, R. Carballedo, A. Perallos, *A Parallel Meta-Heuristic for solving Multiple Asymmetric Traveling Salesman Problem with Simultaneous Pickup and Delivery to solve a Demand Responsive Transport Problem*, Hybrid Artificial Intelligence Systems International Conference, HAIS 2015.

- ◇ E. Osaba, E. Onieva, F. Diaz, R. Carballedo, **P. Lopez-Garcia**, A. Perallos, *A migration strategy for distributed evolutionary algorithms based on stopping non-promising subpopulations: A case study on routing problems*, International Journal of Artificial Intelligence. Esta revista cuenta con un factor de impacto de 1.257 en el Journal Citations Report de 2014.

- ◇ E. Osaba, X.S. Yang, F. Diaz, **P. Lopez-Garcia**, R. Carballedo, *An Improved Discrete Bat Algorithm for Symmetric and Asymmetric Traveling Salesman Problems*, Engineering Applications of Artificial Intelligence. Esta revista cuenta con un factor de impacto de 2.207 en el Journal Citations Report de 2014.

1.4 Estructura del trabajo

La presente memoria se estructura en los siguientes capítulos:

- ◇ El Capítulo 2 presenta un estudio del estado del arte en las diferentes materias de las que se nutre este trabajo, en especial en la Soft Computing y sus componentes, los Sistemas Inteligentes de Transporte, y los tipos de Optimización, así como sus técnicas y funciones.

- ◇ La descripción del algoritmo propuesto en esta tesis, así como su funcionamiento, aportación de la hibridación de las técnicas y aplicación a los FRBS y el resto de problemas se encuentra en el Capítulo 3.
- ◇ En el Capítulo 4 se detallan las diferentes experimentaciones y los resultados obtenidos en cada una de ellas, así como los operadores y codificaciones utilizados por el algoritmo para cada tipo de problema.
- ◇ Por último, las conclusiones extraídas del trabajo realizado y los trabajos futuros que se realizarán tras la terminación de esta tesis se recogen en el Capítulo 5.

Dicen que la curiosidad mató al gato, pero no cuentan si lo que descubrió merecía la pena.

José Saramago

2

Estado del Arte

En este capítulo se recoge el estado del arte en los diferentes ámbitos en los que se desarrolla este trabajo. En la Sección 2.1 se recoge diferente información sobre la Soft Computing, con especial hincapié en metaheurísticas y su hibridación, y Lógica Difusa. Por otro lado, en la Sección 2.2 se encuentran los Sistemas Inteligentes de Transporte, con diferentes definiciones y tipos. Por último, en la Sección 2.3 se definen la optimización de funciones, explicando los tipos que nos podemos encontrar, técnicas que se han aplicado a este tema, y repositorios de funciones benchmark que se encuentran en la literatura actual.

2.1 Soft Computing

Se puede definir la Soft Computing como un conjunto de metodologías cuyo objetivo es explotar la tolerancia al error y la incertidumbre para conseguir manejabilidad, robustez y soluciones de bajo coste y mejores representaciones de la realidad [Zadeh 94].

Podemos encontrarla definida más recientemente en [Verdegay 08], donde la Soft Computing se define como un conjunto de técnicas y métodos que permiten tratar las situaciones prácticas reales de la misma forma que suelen hacerlo los

seres humanos, es decir, en base a inteligencia, sentido común, consideración de analogías, aproximaciones, etc. Se puede considerar a la Soft Computing lo contrario a lo que se denomina Hard Computing.

La Figura 2.1 muestra una diferenciación basándose en los métodos que utilizan tanto la Hard como la Soft Computing. En dicha figura, se muestra como la Hard Computing engloba los métodos precisos, como es el razonamiento inductivo, o la inteligencia artificial, y la optimización numérica tradicional. Estos métodos requieren un modelo analítico y a menudo una gran cantidad de tiempo de cómputo, a la vez que son métodos determinísticos en los que los datos proporcionados deben estar completos. Por su lado, la Soft Computing trata modelos aproximados, jugando a la par con la imprecisión, la incertidumbre y, como su propio nombre indica, la aproximación. Dentro de estos modelos, podemos encontrar los que aplican un razonamiento aproximado, como son la lógica difusa y los modelos probabilísticos, y la aproximación funcional y métodos de optimización, donde encontramos las metaheurísticas y las redes neuronales. A diferencia de los modelos de la Hard Computing, los modelos utilizados por la Soft Computing son estocásticos y capaces de trabajar con datos ambiguos y que, en ocasiones, contienen ruido. A la vez, mientras los métodos de Hard Computing son secuenciales, los de la Soft Computing pueden trabajar en paralelo y sus respuestas son aproximadas, requiriendo normalmente un tiempo de cómputo menor.

Centrándonos en los modelos de la Soft Computing, y más concretamente en los métodos de aproximación/optimización, estos se dividen en metaheurísticas, donde encontramos englobados métodos como el recocido simulado, la búsqueda tabú, colonias de hormigas, optimización por colonia de partículas, búsqueda de vecindarios variables, y la computación evolutiva, basada en métodos biológicos, dentro de la que se ha desarrollado, por ejemplo, la programación evolutiva, las estrategias evolutivas y los algoritmos genéticos. Por otro lado, en la rama del razonamiento aproximado, encontramos la lógica difusa, donde podemos encontrar, por ejemplo, los sistemas basados en reglas difusas. En este trabajo, dentro del amplio abanico de técnicas disponibles, nos centraremos en los algoritmos genéticos y la lógica difusa, de las que hablamos con más detalle en las siguientes secciones.

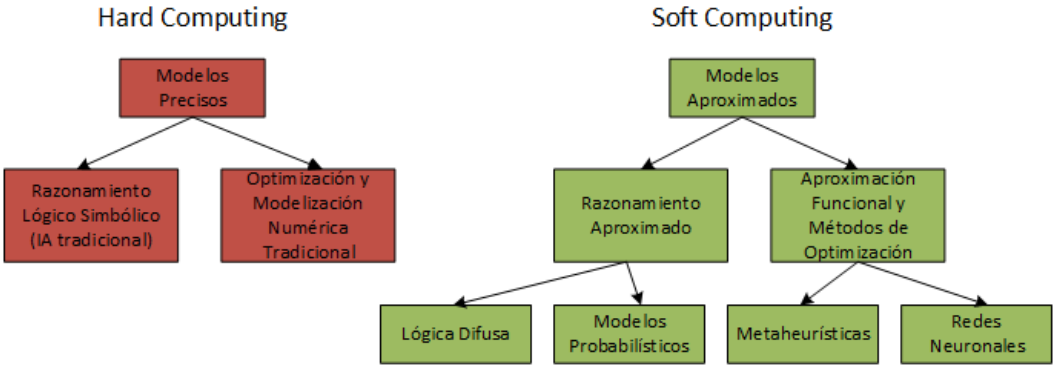

Figura 2.1: Clasificación de técnicas de Hard y Soft Computing

2.1.1 Metaheurísticas

En las últimas décadas, las metaheurísticas han sido ampliamente utilizadas para resolver problemas complejos de optimización. Muchos de estos algoritmos se han inspirado en la naturaleza y han obtenido un gran rendimiento en problemas de alta dimensionalidad. En esta categoría se encuentran algoritmos como PSO [Kennedy 10], Algoritmos Genéticos (GA) [Holland 75], Optimización de colonia de hormigas, del inglés *Ant Colony Optimization* (ACO) [Dorigo 97], Evolución Diferencial, del inglés *Differential Evolution* (DE) [Neri 10] o Recocido Simulado, del inglés *Simulated Annealing* (SA) [Van Laarhoven 87]. Desde su formulación, estos algoritmos se han utilizado en problemas de optimización tales como los presentados en [Cai 10], donde se combinan un SA y una búsqueda local para reducir el coste computacional de la primera técnica minimizando lo posible la temperatura; en [Wang 11], donde se presenta un algoritmo PSO que adopta, bajo una probabilidad, una de cuatro estrategias a la hora de actualizar una partícula, y se aplica a un total de 26 problemas de optimización numérica, cada uno de los cuáles cuenta con características diferentes; en [Ciornei 12], donde la hibridación de un GA y un ACO se aplica a diferentes problemas de optimización continua; o en [Thakur 14], donde un GA se aplica a problemas de optimización multimodal y es comparado con otras implementaciones de GA existentes en la literatura. Además, técnicas probabilísticas como CE [Rubinstein 99] o CMA [Hansen 01] también han sido aplicadas a este tipo de problemas, como se mostrará en secciones posteriores.

2.1.2 Algoritmos Genéticos

Los GA son metaheurísticas de búsqueda que imitan los procesos de selección natural. La idea principal a partir de la cual se define su funcionamiento es mantener una población de cromosomas que representan soluciones candidatas a un problema dado. Dichas soluciones evolucionan con el tiempo mediante un proceso de competición y variación controlada. Cada cromosoma de la población tiene asociado un valor fitness que define la calidad de dicho individuo. Dicha calidad será el factor que que determinará cuáles son los cromosomas utilizados para crear otros nuevos a partir de diferentes operadores.

Desde su aparición en los 70 de la mano de Holland [Holland 75], y gracias a su adaptabilidad a problemas difíciles, han aparecido en la literatura aplicados tanto en solitario [Thakur 14, Osaba 13, Osaba 14] como en combinación con diferentes tipos de técnicas para resolver una gran cantidad de problemas diferentes. En [Adeli 11] se presentan tres sistemas de diagnóstico de enfermedades usando reconocimiento de patrones basado en NN y GA. Por otro lado, se ha utilizado un GA para optimizar las reglas y funciones de pertenencia de dos controladores difusos utilizados para un modelo de control en intersecciones señalizadas presentado en [Qiao 11]. Otro ejemplo es el de [Bevilacqua 12], donde, para optimizar el diseño de redes distribuidas, se utiliza GAs multiobjetivo.

En su versión más clásica, un GA mantiene una población de soluciones candidatas, o individuos. Cuatro procesos son aplicados iterativamente en dicha población, hasta completar un número determinado de ciclos, u otro criterio denominado condición de parada:

1. Operador de selección: Este proceso selecciona dos (o más) individuos para ser padres, dependiendo de su fitness. Los individuos que son mejores que otros, es decir, cuyo valor fitness es el más alto, en caso de estar ante un problema en el que buscamos maximizar el valor retornado por la función de evaluación, o más bajo en caso contrario, tienen una mayor probabilidad de ser seleccionados por el operador. Las características de los padres seleccionados se transferirán a sus descendientes, o hijos, que son creados usando los otros operadores.

2. Operador de cruce: Este operador crea nuevos individuos combinando los padres. Un valor, conocido como probabilidad de cruce (P_{cruce}), está asociado a este tipo de operador. Dicho valor suele encontrarse en el rango [0.5, 1.0] y determina el uso o no de este operador sobre un determinado par de individuos. Esto es, el operador sólo se aplica en el caso de que un número aleatorio supere el valor del parámetro.

3. Operador de mutación: Este proceso modifica un individuo aplicando algún tipo de cambio en su cromosoma. Como el operador de cruce, este operador también tiene un parámetro asociado, conocido como probabilidad de mutación ($P_{mutacion}$). En la mayoría de los casos, este parámetro oscila en el intervalo [0.0, 0.3]. Como en el caso de la probabilidad de cruce, el valor de la probabilidad de mutación determina su aplicación o no sobre el individuo en cuestión.

4. Método de reemplazo: Este proceso define cómo los nuevos individuos reemplazan, total o parcialmente, a los individuos en la población actual. Hay dos tipos de métodos de reemplazamiento. El primero, conocido como modelo generacional, reemplaza toda la población por los hijos creados. El segundo, el modelo estacionario, sólo reemplaza una pequeña parte de la población.

Los GAs han sido utilizados en muchos casos para mejorar o 'afinar' diferentes componentes de un FRBS. En [Onieva 12], un GA es utilizado para obtener la mejor base de reglas posible para un controlador que maneja un vehículo autónomo en intersecciones. En [Cordón 01b], se da una amplia explicación de GAs usados para optimizar diferentes partes de un FRBS. Los GAs y las estrategias evolutivas se utilizan de manera combinada en [Cordón 01a] para aprender a partir de ejemplos bases de datos de tipo Mamdani. En el Algoritmo 1 se encuentra un ejemplo de la estructura que sigue un GA.

2.1.3 Entropía Cruzada

El método de CE fue propuesto por Rubinstein en 1997 [Rubinstein 97]. Fue creado como un método adaptativo para eventos que ocurrieran con probabilidades muy bajas, también llamados *rare-event propabilities*, y para optimización combinatoria.

Datos: $Pob_{tam},\ P_{cruce},\ P_{mutacion}$
Resultado: $Mejor\ individuo\ encontrado$
1 $t \leftarrow 0$
2 $P_0 \leftarrow$ Inicializar Poblacion de Pob_{tam} individuos
3 Evaluar P_0
4 **mientras** $Condicion\ de\ parada\ no\ alcanzada$ **hacer**
5 $Padres \leftarrow$ Seleccionar padres de P_t
6 $Hijos \leftarrow$ Cruce($Padres, P_{cruce}$)
7 $Hijos \leftarrow$ Mutate($Hijos$,$P_{mutacion}$)
8 Evaluar $Hijos$
9 $P_{t+1} \leftarrow$ Reemplazamiento con la Población actual P_t e $Hijos$
10 $t \leftarrow t + 1$
11 **fin**

Algoritmo 1: Pseudocódigo del proceso seguido por un GA.

El nombre de la técnica viene del método de entropía cruzada de Kullback-Leibler [Kullback 51] para medir la cantidad de información necesaria para identificar un evento a partir de un conjunto de probabilidades. CE tiene tres fases principales. las cuáles se ejecutan iterativamente:

1. Generación de un número $Ejemplos_{num}$ de ejemplos aleatorios siguiendo una distribución normal con media $\bar{x}$ y desviación típica σ.

2. Seleccionar un número $Ejemplos_{seleccionados}$ de ejemplos con mejor valor fitness a partir del conjunto creado en la fase anterior.

3. Actualizar los valores de $\bar{x}$ y σ de acuerdo con el valor fitness de los ejemplos escogidos.

CE se puede aplicar en problemas de estimación u optimización [Rubinstein 99], y puede ser usado en diferentes campos. Por ejemplo, en [Haber 10], CE modela un sistema de control difuso para un proceso de perforación. Por otro lado, en [Garcia-Villoria 10], se resuelve usando un CE un problema de programación combinatoria. En [Xia 12] se expone otro caso donde CE se utiliza para un problema de toma de decisiones, ayudando a determinar los pesos óptimos de los atributos.

Conforme el número de iteraciones va avanzando, $\bar{x}$ converge hacia los puntos con mejores resultados mientras que σ disminuye hasta que ambos se centran en el área de mejores soluciones encontradas en el dominio. CE cuenta con un parámetro, $Learn_{rate}$ cuyo valor se encuentra normalmente en el intervalo [0.7, 0.9] [De Boer 05]. Este parámetro se utiliza para actualizar los valores de media y varianza durante la ejecución del algoritmo con las medias y varianzas de los nuevos ejemplos seleccionados.

En el Algoritmo 2 se encuentra el proceso que sigue el método en pseudocódigo. Es importante tener en cuenta que el algoritmo está presentado para un problema unidimensional; en el caso de contar con más dimensiones, $\bar{x}$ y σ deben ser vectores y cada una de dichas dimensiones debe ser tratada de manera independiente al resto.

Datos: $Ejemplos_{num}, Ejemplos_{actualizar}, Learn_{rate}$
Resultado: $Mejor\ individuo\ encontrado$
1 $\overline{x} \leftarrow$ Inicializar Medias
2 $\sigma \leftarrow$ Inicializar Desviaciones
3 **mientras** $Condicion\ de\ parada\ no\ alcanzada$ **hacer**
4 $Ejemplos \leftarrow$ Generar $Ejemplos_{num}$ Ejemplos con distribución normal N($\overline{x}$,σ)
5 Evaluar $Ejemplos$
6 $Ejemplos_{seleccionados} \leftarrow$ Seleccionar los $Ejemplos_{actualizar}$ mejores individuos de $Ejemplos$
7 $\overline{x} \leftarrow (1 - Learn_{rate}) \cdot \overline{x} + Learn_{rate} \cdot$ Media($Ejemplos_{seleccionados}$)
8 $\sigma \leftarrow (1 - Learn_{rate}) \cdot \sigma + Learn_{rate} \cdot$ Varianza($Ejemplos_{seleccionados}$)
9 **fin**

Algoritmo 2: Pseudocódigo del proceso seguido por la Entropía Cruzada

2.1.4 Hibridación de técnicas

A pesar de su éxito en muchos y variados problemas y el grandísimo número de técnicas existentes, las metaheurísticas aún tienen varios problemas a los que no se les ha encontrado una solución por el momento. Uno de ellos es la variabilidad de

su rendimiento cuando se escoge un mismo algoritmo para diferentes problemas, ya que dependen de las características de la función a optimizar. Otro, relacionado con el anterior, es la selección del algoritmo apropiado para cada tipo de problema. Centrando nuestra atención en el primer problema, uno de los enfoques tomados para abordarlo es la combinación de dos o más algoritmos para obtener uno mejor o contrarrestar sus deficiencias. Esta combinación es llamada Hibridación [Topcuoglu 07].

La hibridación de técnicas es un tema importante en la literatura [Purwar 15, Hernández 13]. El concepto se basa en combinar dos o más técnicas para crear sinergia entre ellas y mejorar su rendimiento, además de disminuir o eliminar sus carencias. Las técnicas utilizadas obtienen un buen rendimiento para resolver un problema específico, y su combinación hace que se mejoren los resultados obtenidos si se hubieran ejecutado en solitario cada una de sus partes. Los algoritmos híbridos han demostrado ser prometedores en muchos campos en general. Algunos ejemplos son, entre otros, [Purwar 15], donde se combina la técnica de clúster K-means con un perceptrón multicapa para crear un modelo de predicción. Dicho modelo elige una técnica entre 11 propuestas para su utilización en datos médicos incompletos (missing values). [Sangsawang 14] propone dos enfoques híbridos, un GA con autoajuste adaptativo y un PSO con distribución Cauchy para un problema con restricciones y los compara con metaheurísticas clásicas. Por último, se puede encontrar un survey reciente sobre la hibridación, ya no sólo de metaheurísticas entre sí, sino con otros tipos de técnicas como los algoritmos exactos, en [Blum 11]. En el campo de la optimización en particular podemos encontrar también variedad de hibridaciones. Algunos ejemplos son los presentados en [Abd-El-Wahed 11], donde se combinan las técnicas PSO y GA para actualizar las partículas del primer método, usando dicha hibridación en funciones multimodales extraídas de la literatura. También contamos con [Hernández 13], en el cual se presenta un método que combina DE con un algoritmo Hill Climbing para optimización con restricciones. Por último, en [Mandal 11] se puede encontrar DE, en este caso combinado con un algoritmo de búsqueda local, que se aplica posteriormente a problemas de optimización del mundo real.

El algoritmo que ha sido desarrollada durante esta tesis se basa en la combinación de dos de las técnicas nombradas anteriormente, un GA y una CE.

2.1.5 Lógica Difusa

La lógica difusa, introducida por Zadeh en 1965 [Zadeh 65], permite procesar información imprecisa usando reglas del tipo IF-THEN. Surge de la necesidad de controlar sistemas de los que no se tiene más que descripciones lingüísticas, incompletas e inexactas basadas muchas veces en apreciaciones subjetivas de las variables de control [Zadeh 96]. La lógica difusa se basa en el concepto de conjunto difuso, donde la idea principal es que los elementos tengan un grado de pertenencia a un conjunto. Dicho grado de pertenencia se define en el intervalo $[0, 1]$, en vez de únicamente un valor $\{0, 1\}$.

El grado de pertenencia de una variable a un conjunto viene dado por su función de pertenencia. Una función de pertenencia de un conjunto difuso es una generalización de una función característica en un conjunto clásico. Representa el grado con el que un elemento pertenece a un conjunto. A la hora de determinar una función de pertenencia, se eligen funciones sencillas. Contamos pues con diferentes tipos, siendo las más comunes la función triangular, función gamma, función S, Gaussiana, y trapezoidal, entre otras. Centrándonos en las funciones triangulares y trapezoidales, siendo éstas últimas las utilizadas en el trabajo que nos ocupa, podemos decir que las funciones triangulares son aquellas que cuentan con tres puntos característicos, y, como indica su nombre, la función tiene forma de triángulo. La Ecuación 2.1 muestra la formulación matemática de la misma.

$$A = \begin{cases} (x-a)/(m-a) & \text{si } x \in (a, m] \\ (b-x)/(b-m) & \text{si } x \in (m, b) \\ 0 & \text{en cualquier otro caso} \end{cases} \tag{2.1}$$

siendo a el punto donde comienza la función, m el punto donde la función de pertenencia es igual a 1, y b el punto donde finaliza la función. Por otro lado, contamos con las funciones trapezoidales, las cuales cuentan con cuatro puntos. Se puede decir que estas funciones son una extensión de las triangulares ya que el punto extra con el que cuentan es el punto medio m que se transforma en un intervalo. La Ecuación 2.2 muestra la codificación de este tipo de funciones.

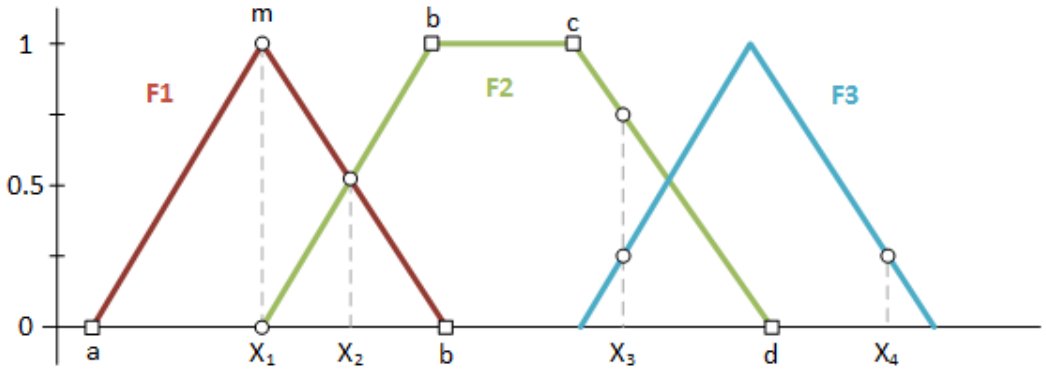

Figura 2.2: Ejemplo de funciones de pertenencia triangulares y trapezoidales

$$A = \begin{cases} (x-a)/(b-a) & \text{si } x \in (a,b] \\ 1 & \text{si } x \in (b,c) \\ (d-x)/(d-c) & \text{si } x \in (b,d) \\ 0 & \text{en cualquier otro caso} \end{cases} \tag{2.2}$$

donde a es el punto donde comienza la función, b y c son los puntos entre los cuáles la función de pertenencia toma valor 1, y d el punto donde ésta finaliza. Con la idea de ilustrar ambos tipos de funciones y dejar todo lo claro posible los puntos que las delimitan, un ejemplo gráfico se puede encontrar en la Figura 2.2 con varias funciones triangulares y una función trapezoidal.

Las funciones F1 y F3 son triangulares mientras que la función F2 es trapezoidal. Contamos en dicha gráfica con cuatro elementos, con distintos grados de pertenencia a dichas funciones. Por ejemplo, el elemento x_1 tendrá un grado de pertenencia 1 a la función F1 y un grado de pertenencia 0 a las funciones F2 y F3. Además, es el punto a de la función F2. x_2 cuenta con grado de pertenencia 0.5 a las funciones F1 y F2, mientras que x_3 tiene un grado de pertenencia 0.75 a la función F2 y 0.25 a la función F3. Por último, x_4 cuenta con un grado de pertenencia 0.25 a la función F3. De esta manera, se muestra como un elemento puede tener grados de pertenencia a dos o más clases diferentes (elementos x_1, x_2 y x_3), y como éste puede tener el mismo valor pero ser elementos completamente diferentes (elementos x_3 y x_4)

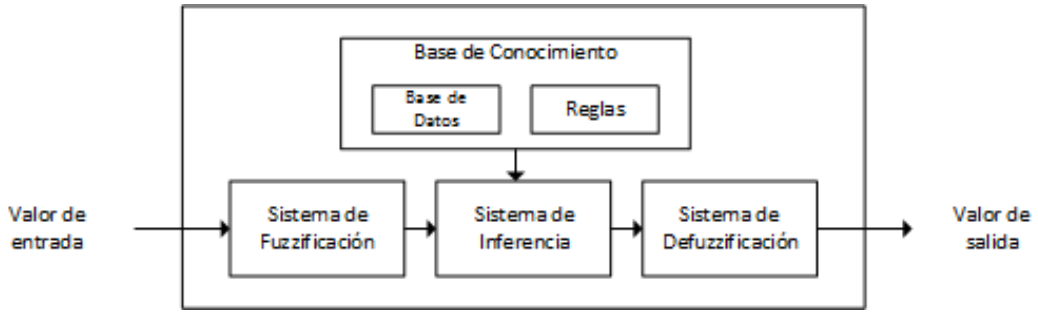

Figura 2.3: Ejemplo de un sistema basado en reglas difusas

En problemas de tráfico, como en [Onieva 09] o en [Lopez-Garcia 16a], la lógica difusa se ha utilizado a menudo dado que permite que la información y las decisiones puedan ser tratadas con este tipo de reglas. Por ejemplo: *si la ocupación es alta y el flujo de coches es bajo, entonces existe congestión en la vía.*

Uno de los tipos de sistemas difusos más utilizado son los FRBS. Podemos definirlos como una extensión de los sistemas basados en reglas, dado que trabajan con reglas *IF-THEN* cuyos antecedentes y consecuentes están compuestos por reglas de lógica difusa, en vez de lógica clásica. Estos sistemas están formados por varias entradas, un sistema de inferencia difuso con un conjunto de reglas en su interior y una salida. Un ejemplo de esta clase de sistemas se encuentra en la Figura 2.3

Los FRBSs son utilizados en diferentes tipos de problemas de la vida real. En [Ghorbani 10], se utiliza uno de estos sistemas para simular diferentes políticas de gestión de energía en vehículos eléctricos. Clasificadores basados en reglas difusas y sistemas envolventes difusos son usados en [Iglesias 10] para reconocer acciones o actividades de los usuarios para predecir futuras acciones y monitorizar la salud de dichos usuarios remotamente. Podemos encontrar otro ejemplo en [Awan 11], donde se utiliza un FRBS para predecir el tiempo atmosférico.

En la literatura se distinguen, principalmente, dos tipos de FRBS en función del tipo de consecuente:

1. FRBS de tipo Mamdani. Este tipo de sistemas utiliza variables de entrada y salida reales. Consta de un sistema de fuzzificación, que transforma dichas variables a su valor en los conjuntos difusos, un sistema de inferencia, la base de reglas, que contiene la información sobre el problema en forma de

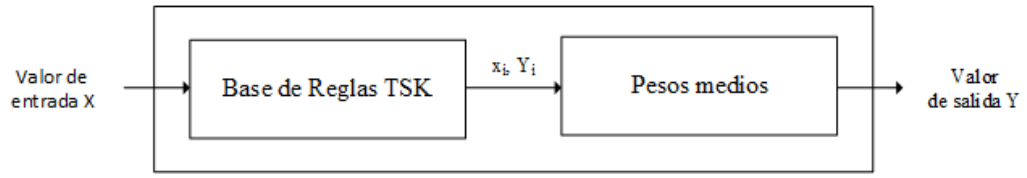

Figura 2.4: Ejemplo de un sistema de tipo TSK

reglas *IF-THEN*, y un sistema de defuzzificación, que vuelve a transformar los valores a su valor real. El resultado de la regla es, por tanto, un conjunto difuso. Un ejemplo sería "Si X_1 es ALTO y X_2 es MEDIO entonces Y es ALTO". La Figura 2.3 muestra un ejemplo de este tipo de sistema.

2. FRBS de tipo TSK (Takagi-Sugeno-Kang). En este caso, en vez de trabajar con reglas lingüísticas, se proponen reglas cuyo antecedente está compuesto por variables lingüísticas y el consecuente se representa como una función utilizando las variables de entrada. El resultado de la regla es, por tanto, una función. Siguiendo con el ejemplo anterior, "Si X_1 es ALTO y X_2 es MEDIO entonces Y es f(X_1,X_2)". De una manera más formal, la salida de un sistema TSK se obtiene mediante la media ponderada de las salidas individuales provistas por cada regla Y_i, y la entrada del sistema. La Figura 2.4 ilustra este proceso, a la vez que muestra la diferencia entre un sistema como el presentado en la Figura 2.3 y el actual, siendo dicha diferencia, como se ha dicho, la base de reglas utilizada y la inclusión de pesos medios para el cálculo de la salida.

En secciones posteriores se hablará más detalladamente de la composición y aplicación de los sistemas TSK al trabajo desarrollado.

2.2 Sistemas Inteligentes de Transporte

Los ITS son el vínculo entre las tecnologías de la información y la comunicación y los vehículos y redes que transportan personas o mercancías.

Hay definiciones muy diversas realizadas por diferentes organismos e instituciones de gran relevancia. Por ejemplo, el Departamento de Transporte de EEUU lo define como:

Los ITS consisten en la aplicación de la tecnología avanzada de computación, electrónica y comunicaciones para incrementar la seguridad y la eficiencia del transporte de superficie.

Por otro lado, el Ministerio de Fomento Español lo define como:

Los ITS se pueden definir como un conjunto de aplicaciones avanzadas dentro de la tecnología informática, electrónica y de comunicaciones que, desde un punto de vista social, económico y medioambiental, están destinadas a mejorar la movilidad, seguridad y productividad del transporte, optimizando la utilización de las infraestructuras existentes, aumentando la eficiencia del consumo de energía y mejorando la capacidad del sistema de transportes.

Los ITS surgen a finales de los 80 y principios de los 90 como alternativa sostenible al problema generado por la creciente demanda de movilidad, especialmente en ámbitos urbanos e interurbanos.

Las áreas donde se aplican estos sistemas son, principalmente, las siguientes:

- ◇ Sistemas de Tratamiento del Tráfico Avanzado: Integraría el tratamiento de diferentes funciones de carretera. En este apartado se encontraría, por ejemplo, la predicción de las posibles retenciones de tráfico y las instrucciones que se darían a los vehículos de manera que se mejorara la eficiencia de la red de la carretera. Una de las claves de este apartado sería la detección de los incidentes de forma que se redujera la congestión que provocan, y posibilitarían una actuación más rápida.

- ◇ Sistemas de Información al Viajero: Provee al viajero de información en sus vehículos, hogares o lugares de trabajo. Esta información incluye: localización de incidentes, información meteorológica, rutas óptimas, etc.

- ◇ Sistemas de Control de Vehículo: Se busca en estos sistemas mejorar la seguridad durante el viaje y la eficiencia del vehículo. En este área se englobarían los sistemas de alerta de colisión, tanto automáticos (el coche activaría el freno automáticamente) como manuales (dependiendo del usuario), o la información y control de la infraestructura, como por ejemplo el manejo de la

velocidad de los vehículos mientras van por una vía que tiene establecida una velocidad determinada.

- ◇ Operaciones con Vehículos Comerciales: Los sistemas que se encuentran en esta categoría buscan mejorar la productividad y eficiencia de las flotas de camiones, furgonetas, y taxis.

- ◇ Sistemas de Transporte Público: En este punto se encuentran las soluciones que se aportan para mejorar la accesibilidad a la información de los usuarios del transporte público así como la mejora del horario que este mantiene y la utilización de las flotas de vehículos.

- ◇ Sistemas de Transporte Rural: Esta opción es un reto para los ITS debido a las restricciones económicas relativas a las carreteras con poca densidad de vehículos en muchos lugares rurales.

Por otro lado, las ITS intervienen en los diferentes tipos de transporte de maneras variadas:

- ◇ El transporte aéreo es uno de los tipos en los que más se ha avanzado, debido a la necesidad de ordenar el espacio aéreo con sistemas de localización, control de vuelo, o en las relaciones entre aviones y aeropuertos.

- ◇ En el caso del ferrocarril, siendo la seguridad uno de los objetivos más importantes, las circulaciones se han regulado utilizando los últimos avances tecnológicos, apoyándose en su singularidad de interacción contínua entre los vehículos y la vía.

- ◇ Para transporte marítimo, los avances tecnológicos se han centrado en las telecomunicaciones, ya sea a la hora de navegar o de usar localización en un área determinada.

- ◇ Por último, el transporte de carretera es actualmente un foco de actuación constante que avanza a pasos agigantados dado la cantidad de problemas que presenta, siendo algunos de los más importantes la congestión en carretera, o la accidentalidad creciente.

Centramos pues lo que resta de esta sección en el transporte por carretera, donde se encuentra englobado la parte principal del trabajo realizado en esta tesis. Como se ha dicho anteriormente, las razones impulsoras del desarrollo de los ITS en este campo son la accidentalidad, las congestiones de tráfico y el deterioro del medio ambiente, por lo que los avances se han centrado en temas como la seguridad, la capacidad de las infraestructuras y los efectos contaminantes de los vehículos.

Podemos encontrar dos áreas:

1. **Infraestructuras inteligentes**, cuya finalidad es mejorar la seguridad del transporte público y privado desde el entorno de circulación, ya sea proporcionando instalaciones o servicios más eficientes. Un ejemplo podría ser sistemas de gestión de emergencias, de información meteorológica o información de tráfico y viaje.

2. **Vehículos Inteligentes**, mejorando la seguridad y movilidad de los vehículos instalando sistemas como sensores, equipos informáticos y de comunicaciones, que mitigarían las consecuencias de los accidentes que pudieran ocurrir. Áreas de interés dentro de este campo son los vehículos autónomos o los sistemas de asistencia a la conducción.

Un diagrama donde se resume la información de esta sección se encuentra en la Figura 2.5. En nuestro caso, donde se quiere predecir la congestión en una vía, y tal y como queda recalcado en el diagrama antes mencionado, el trabajo pertenecería al área de infraestructuras inteligentes, y más concretamente, a los sistemas de tratamiento del tráfico.

2.3 Optimización

A lo largo de esta sección, se describen los diferentes tipos de optimización que se pueden encontrar en la literatura en la Sección 2.3.1, un estudio de las diferentes técnicas de Soft Computing aplicadas a los diferentes tipos presentados en la Sección 2.3.2, y, por último, diferentes tipos de funciones benchmark que se usan en diferentes artículos, las funciones que contienen y sus diferentes características en la Sección 2.3.3.

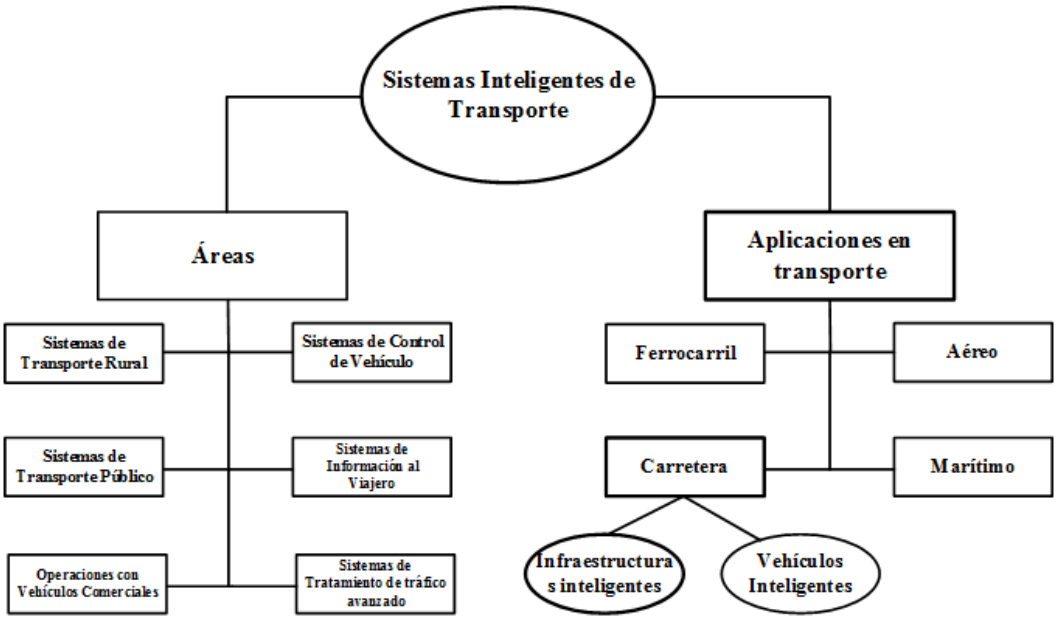

Figura 2.5: Diagrama resumen de los Sistemas Inteligentes de Transporte, sus áreas y sus aplicaciones

2.3.1 Tipos de Optimización

En secciones anteriores se ha hablado brevemente de los tipos de optimización, siendo los más destacados la optimización numérica [Schwefel 81], lineal [Bertsimas 97], continua [Eiselt 87] o combinatoria [Papadimitriou 98].

Adentrándonos más en cada una de ellas, podemos definir la optimización numérica, también llamada solamente optimización, de la siguiente forma: Considerando una función $f(x)$, se busca un valor x por el cual la función f obtiene su valor máximo o mínimo y, tal que $Max/Min\ y = f(x)$.

Ambas variables x e y pueden ser tanto un solo valor, como un vector de ellos. La función f puede tener una serie de restricciones que se deben satisfacer, pudiendo diferenciar así optimización con o sin restricciones. Este es, por tanto, el caso más básico de optimización que nos podemos encontrar en la literatura [Nocedal 06].

En el caso de la optimización lineal, o programación lineal, el objetivo general es minimizar una función objetivo lineal de variables reales continuas sujetas a restricciones lineales. Podemos contar con dos tipos de soluciones: una solución

viable, es decir, aquella que satisface todas las restricciones, o una solución óptima, que a su vez sería viable, siendo esta aquella que obtiene el menor valor de la función objetivo. Para resolver este tipo de problemas podemos usar dos métodos: métodos simples [Lagarias 98], que buscan reducir este tipo de problemas a sistemas cuadráticos, de manera que puedan ser resueltos por valores únicos, o los llamados métodos *barrera* o de punto interior [Wright 05], derivadas de la programación no lineal. Un ejemplo de esta clase de optimización podemos encontrarlo en [Murat 14], donde se busca minimizar el tiempo de acceso de los viajeros y el tiempo de viaje de las líneas de autobús de la ciudad de Izmir, en Turquía. Otro ejemplo lo podemos encontrar en [Bouras 13], donde se resuelve un problema lineal de un sólo objetivo con una combinación de técnicas. Por último, se presenta un algoritmo híbrido entre un GA y un PSO para resolver un problema lineal de dos niveles en [Kuo 15].

En la optimización combinatoria, se tienen una familia de subconjuntos que forman parte de un conjunto finito, tales que el objetivo es encontrar uno de ellos que satisfaga la función objetivo. Ejemplos de este tipo de optimización pueden ser el problema del viajante de comercio (o Traveling Salesman Problem (TSP)) [Osaba 14, Zhang 12a] o problemas de empaquetado [He 13, Silva 14].

Centrándonos en la optimización continua, las variables toman valores dentro del dominio de los números reales [Wright 99]. Esta característica distingue este tipo de optimización de otras como la combinatoria o la discreta, donde las variables pueden ser binarias, enteros u objetos de un conjunto. Se han aplicado algoritmos iterativos a la optimización continua, los cuales generan una secuencia de valores en cada iteración para, finalmente, converger en una solución válida. Debido a su alta complejidad, diferentes técnicas han sido aplicadas a esta clase de problemas [Jourdan 09, Elsheikh 14, Holland 75], entre ellas, técnicas de inteligencia artificial.

Brevemente, se nombran a continuación otros tipos de optimización, dependiendo de diversos factores como:

- ◇ Restricciones: podemos contar con problemas sin restricciones o con restricciones. Los problemas con restricciones se pueden tratar como uno sin ellas

si se sustituyen dichas restricciones por valores de penalización en las funciones objetivo.

- ◇ Objetivos: la mayoría de los problemas de optimización tienen una única función objetivo. Se pueden tener también con muchas funciones objetivo, o con ninguna. En este último caso, el problema se transforma en la búsqueda de valores que cumplan las restricciones propuestas, creando así soluciones viables.

- ◇ Datos: dependiendo del conocimiento o no de todos los datos que proporciona el problema, podemos contar con modelos determinísticos, donde los datos se conocen a ciencia cierta, o modelos estocásticos, que trabajan con la incertidumbre y, por tanto, con distribuciones de probabilidad.

2.3.2 Técnicas de Soft Computing aplicadas a Optimización

La literatura está llena de ejemplos de todos los tipos anteriormente mencionados, demostrando de esta manera que la optimización es un campo en continuo crecimiento y avance. Usando la base de artículos Scopus como referencia, a lo largo del año 2014, es decir, desde el 1 de Enero hasta el 31 de Diciembre, un total de 6139 artículos contienen uno de los tipos de optimización de los que se ha hablado en el apartado anterior entre las keywords del documento. Sobre esta cantidad, alrededor de un 80 % de los artículos contienen estudios sobre optimización numérica y lineal (40 % para cada uno , con 2471 y 2474 artículos respectivamente), mientras que la optimización combinatoria cuenta con un 11 % (678 artículos) y la optimización continua el restante 9 % (516 artículos). La Figura 2.6 contiene cuántos documentos de cada clase hay en dicho buscador.

En todos los casos, podemos encontrar técnicas de Inteligencia Artificial, concretamente de Soft Computing, aplicadas a esta temática. Para optimización lineal, podemos encontrar [Valizadeh 14], donde se aplica un modelo lineal para la cadena de suministro de biofuel en Malasia. Sobre dicho modelo se utiliza un PSO multiobjetivo y se compara con un algoritmo genético multiobjetivo del estado del arte llamado NSGA-II [Deb 02]. Otro ejemplo lo tenemos en [Azar 14] donde, entre todas las SVM que se utilizan para el diagnóstico del cancer de mama, se encuentra

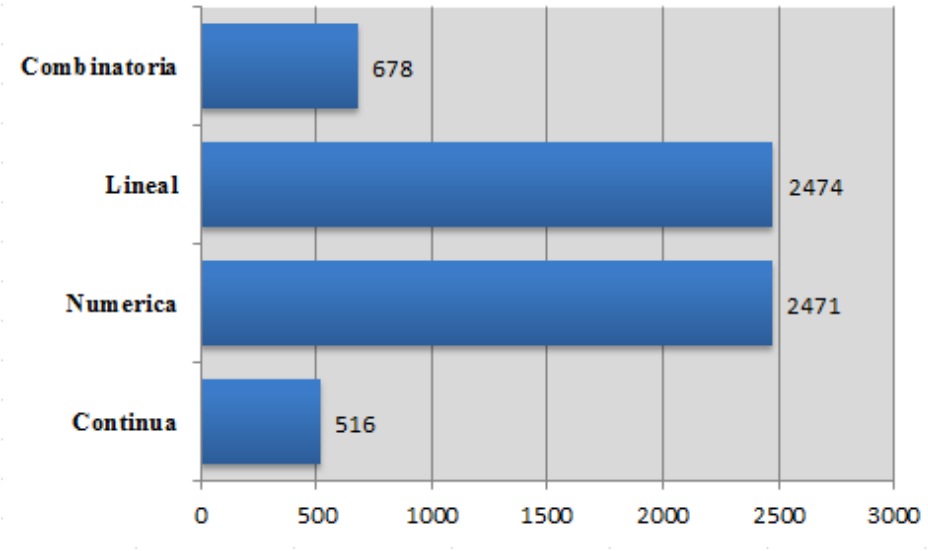

Figura 2.6: Artículos publicados durante el año 2014 en cada tipo de optimización (fuente: Scopus)

una basada en la programación lineal, la cual obtiene un más de un 97 % de acierto en el dataset utilizado.

Como se ha mencionado en el apartado anterior, hay ciertos problemas muy conocidos y utilizados en la literatura sobre optimización combinatoria. Uno de ellos es el TSP. La Soft Computing se ha utilizado mucho en este tipo de problemas, por ejemplo, en [Osaba 14], donde se aplica una metaheurística creada por el autor para resolver problemas de este tipo, o en [Okulewicz 13], donde se aplica un PSO a diferentes tipos de problemas de optimización combinatoria, entre ellos el TSP, pero además un problema de empaquetamiento (problema de la mochila).

Para optimización continua encontramos [Liao 14], donde se propone una ACO unificado para este tipo de optimización. Dicha propuesta combina tres variantes de este tipo presentados previamente. La propuesta es aplicada a funciones benchmark utilizadas en diferentes congresos y revistas. Un algoritmo ACO se aplica también para la la resolución de problemas de optimización continua en [Chen 14]. En [Toledo 14] se utiliza un GA con estructura de población jerárquica para resolver problemas continuos sin restricciones. Tanto este como un SA, DE y un PSO se utilizan en funciones benchmark en dicho artículo.

2.3.3 Funciones Benchmark

En el apartado anterior, algunos de los artículos mencionados han utilizado funciones benchmark extraídas o bien de revistas que las proporcionan, de congresos que cuentan con workshops dedicados a la temática de la optimización, o simplemente de internet. Esta sección se encarga de recoger algunas de esas recopilaciones de funciones, explicar sus características y para qué tipo de optimización se podrían utilizar dichas funciones.

Según el teorema *no free lunch* enunciado en [Wolpert 97], si se comparan dos algoritmos con todas las posibles funciones de un conjunto, el rendimiento de ambos algoritmos sería, de media, el mismo. Por lo tanto, tener un conjunto de funciones extremadamente grande, donde se recoja todas las posibles es, en esencia, algo infructuoso. Por ello, cuando se evalua un algoritmo, se debe buscar en qué tipo de problemas obtiene mejor rendimiento. Para esta tarea, debemos hacer un estudio de pocas pero bien elegidas funciones de diferentes tipos [Whitley 95]. Así, podríamos obtener conclusiones del rendimiento del algoritmo según el tipo de función en el que se aplica. Una función benchmark es, por tanto, una de las funciones que forman el conjunto a evaluar. La literatura cuenta con varios ejemplos de conjuntos de funciones, como pueden ser las extraídas de la tesis de De Jong [De Jong 75] o las que se pueden encontrar en el artículo [Eiben 97]. Por otro lado, la revista Soft Computing publicó un número especial dedicado a problemas de optimización continua de gran escalabilidad donde se recogían un total de 19 funciones [Lozano 11]. Entre estas funciones, se cuenta con las expuestas en el congreso dedicado a la computación evolutiva IEEE Congress on Evolutionary Computation (CEC) en una competición sobre este tipo de problemas en el año 2005. En el último año de esta competición (en el momento en el que se escribe este documento, 2015), cuenta con un total de 15 funciones de 1000 dimensiones cada una. Tanto en esta conferencia, como en la célebre Genetic and Evolutionary Computation Conference (GECCO), se realizan competiciones de optimización utilizando diferentes conjuntos de funciones, dependiendo de la competición, cuyos resultados y contenido son totalmente accesibles para aquellos que quieran utilizarlas. El algoritmo presentado en esta tesis participó en la Black-Box Optimization Benchmarking (BBOB) celebrada en la conferencia GECCO'15. En esta competición,

Nombre	Nº de Funciones	Tipos de Funciones	Referencia
De Jong	5	-	[De Jong 75]
Eiben	7	Unimodales, Multimodales, Separables	[Eiben 97]
SOCO	19	Escalables	[Lozano 11]
CEC	15	4 (ver Referencia)	Referencia [3]
BBOB	24	5 (ver Referencia)	Referencia [4]
Mittelmann	-	-	Referencia [5]

Tabla 2.1: Tipos de benchmark encontrados en la literatura y sus características

se proponían un total de 24 funciones de cinco tipos diferentes[1]: separables, con condicionamiento bajo o moderado, con condicionamiento elevado y unimodal, multimodales con estructura global adecuada, y multimodales con estructura global débil. Los resultados obtenidos con las funciones utilizadas en esta competición se mostrarán en capítulos siguientes.

Por último, diferentes páginas ofrecen benchmark para los diferentes tipos de optimización explicados en los apartados anteriores. Un ejemplo puede ser el creado por Hans Mittelmann en [2], donde se pueden encontrar actualizados diferentes benchmark a elegir.

Con la intención de resumir todo lo recogido en esta sección, en la Tabla 2.1 se encuentran las diferentes opciones propuestas.

[1] http://coco.gforge.inria.fr/doku.php?id=bbob-2015
[2] http://plato.la.asu.edu/bench.html
[3] http://staff.ustc.edu.cn/ ketang/lsgo2015.html
[4] http://coco.gforge.inria.fr/doku.php?id=bbob-2015
[5] http://plato.la.asu.edu/bench.html

No importa lo rápido que viaje la luz, siempre se encuentra con que la oscuridad ha llegado antes y la está esperando.

Terry Pratchett

3

Descripción del algoritmo

En este capítulo se tratarán todos los aspectos relacionados con el algoritmo que se presenta en este trabajo. La explicación y funcionamiento del método se encuentra en la Sección 3.1. Por otro lado, su aportación a los algoritmos híbridos, así como el área en el que se encuentra ubicado se muestra en la Sección 3.2. En la Sección 3.3 se habla de la aplicación del método a la optimización de los FRBS, mientras que en la Sección 3.4 se explica su aportación al campo de la optimización de funciones. Por último, en la Sección 3.5 se explican algunos de los detalles que hay que tener en cuenta a la hora de aplicar el algoritmo.

3.1 GACE: Explicación y Funcionamiento

Se han desarrollado una gran cantidad de metaheurísticas a lo largo de la última década, como se ha expuesto en el estado del arte. Sin embargo, pese a su éxito en diferentes problemas, todavía sufren de varios problemas que se consideran aún abiertos, ya sea a la hora de elegir qué técnica es mejor para qué problema, o las debilidades de cada una de ellas. Centrándonos en las últimas, y poniendo como ejemplo metaheurísticas poblacionales como GA y ACO, cuentan con problemas

de explotación del espacio de búsqueda [Talbi 02]. Otro ejemplo, teniendo en cuenta en este caso los algoritmos basados en trayectoria, como pueden ser el SA y la Búsqueda Tabú, es la posibilidad de quedar estancados en un óptimo local debido a su mal comportamiento en lo que a exploración del espacio de búsqueda se refiere [Wang 04]. Por último, otro problema, llamado el problema de Selección de Algoritmo, retrata el hecho de que, al contar con tantos algoritmos disponibles, no sea una tarea sencilla saber cuál obtendrá mejor resultado con la información disponible [Muñoz 13].

El algoritmo GACE, llamado de esta manera por combinar un Algorito Genético y un método de Entropía Cruzada (Genetic Algorithm with Cross Entropy en inglés) está diseñado con la idea de aprovechar la habilidad de exploración de un GA y la habilidad de explotación de CE en un problema de optimización dado. La principal aplicación dada, y en la que se centra el desarrollo de esta tesis, es la optimización de una jerarquía de sistemas difusos con el objetivo de mejorar las entradas, etiquetas y reglas de cada uno de los sistemas que lo forman. Por otro lado, el algoritmo se ha aplicado a otros campos para probar su rendimiento además de demostrar su capacidad de adaptación a diferentes problemas que se le pueden plantear como se verá en secciones posteriores. El hecho de usar estos algoritmos hace que se intente evitar el estancamiento por parte de CE en un óptimo local gracias a la habilidad de exploración del GA, mientras que se explota todo lo posible el espacio de búsqueda por la aplicación del CE, evitando de esta manera el problema presentado en el párrafo anterior sobre los algoritmos poblacionales.

Sabiendo pues los motivos de desarrollo de este método híbrido, se pasa a explicar de manera detallada su funcionamiento general, que se seguirá sea cual sea el problema al que se aplique. En primer lugar, se inicializará la población de soluciones. Dependiendo del problema que se esté tratando, esta inicialización se realizará de manera distinta, ya que depende de la codificación de los individuos para dicho problema. Aún así, un punto común de todos los problemas es que cada elemento de la población se inicializará de forma completamente aleatoria. Una vez realizado este paso, la población se divide en dos subpoblaciones:

1. Población dedicada al GA, que contará con individuos a los que se les aplicará los diferentes operadores de un GA. Estos individuos serán elegidos por

el respectivo operador de selección escogido.

2. Población dedicada al CE, con un número de individuos que se utilizarán en una CE. Los individuos que forman esta subpoblación son los mejores individuos existentes en la población actual de soluciones.

Los tamaños de subpoblación son elegidos por el usuario y su suma debe ser igual al tamaño de la población actual.

Una vez que ambas poblaciones se han escogido, cada una se usa de manera diferente:

◇ En la población dedicada al GA se aplican los operadores de cruce y mutación para generar los nuevos individuos. Los operadores concretos a utilizar variarán según el tipo de problema a abordar y se especifican en la Sección 4.1.3.1.

◇ En el caso de la subpoblación dedicada al CE, se aplica el método CE a sus individuos. Primero se actualizan las medias y desviaciones empleando el Algoritmo 2 (líneas 7 y 8). Tras esto, los individuos son generados aleatoriamente siguiendo una distribución normal usando dicha media y desviación actualizadas. En la Sección 4.1.3.2 se encuentra este proceso aplicado, en este caso, al problema de predicción de congestión y sus individuos.

Cuando ambos algoritmos son aplicados a sus respectivas subpoblaciones, con los individuos resultantes de éstas se crea una población completamente nueva que contendrá, por tanto, individuos de los operadores del GA e individuos del método CE. Esta nueva población reemplaza en su totalidad a la anterior, por lo que GACE es un algoritmo generacional. Se aplica elitismo en la nueva población, es decir, se mantiene en la población al mejor individuo encontrado en la anterior generación. La Figura 3.1 muestra el proceso descrito.

Cabe destacar que los individuos pueden formar parte de ambas poblaciones a la vez. De esta manera se consigue que sean los mejores individuos los que vayan 'mejorando'con el paso de las generaciones, además de terminar centrando la mejora en un espacio de búsqueda concreto a lo largo de la ejecución del algoritmo.

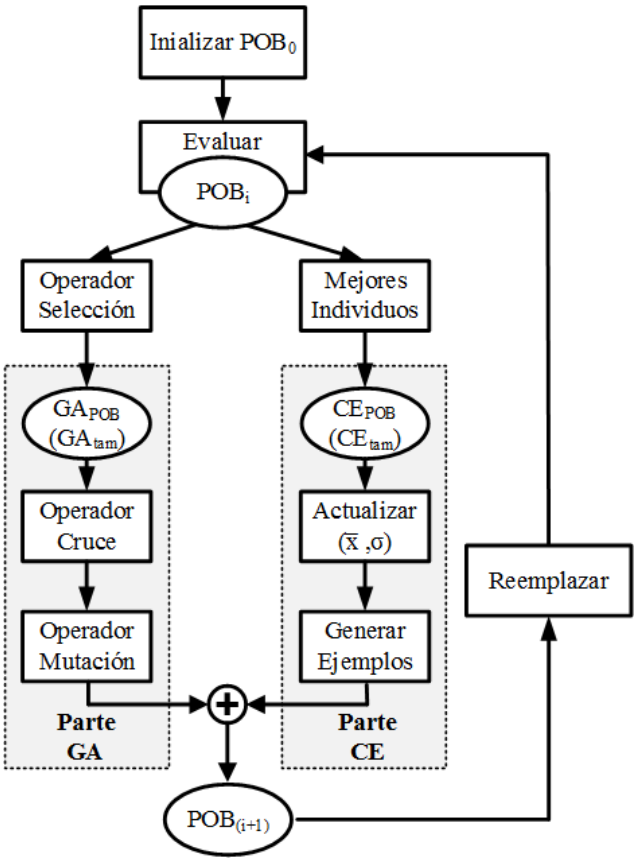

Figura 3.1: Pasos que sigue el algoritmo GACE

3.2 Combinación de las técnicas: Clasificación y aportación

Tras haber realizado una explicación del funcionamiento del algoritmo, y habiendo realizado anteriormente una explicación detallada de la aplicación y uso de los Algoritmos Genéticos y la Entropía Cruzada en los diferentes temas de los que trata esta tesis, surge la necesidad de especificar las diferentes categorías, siempre siguiendo una taxonomía detallada, en las que residiría el algoritmo desarrollado según las técnicas encontradas en el estado del arte.

Esta clasificación surge debido a cómo se ha extendido durante la última década la hibridación de diferentes algoritmos de manera que se cree una sinergia entre ellos, mayoritariamente para suplir las debilidades de uno de los métodos con me-

canismos o características de otro, y viceversa. Ejemplos de estas debilidades son, por ejemplo, las dadas en el capítulo anterior: la falta de explotación del espacio de búsqueda por parte de metaheurísticas basadas en poblaciones [Talbi 02] o, en el caso de algoritmos basados en trayectorias, la facilidad con la que pueden quedarse atrapados en óptimos locales [Wang 04]. En el caso de algoritmos probabilísticos como DE, se han identificado problemas como la forma específica en la que se crean los nuevos individuos o el potencial de generar sólo un número limitado de soluciones en una generación [Segura 15].

Se pueden encontrar diferentes taxonomías en la literatura que buscan clasificar los algoritmos híbridos siguiendo una serie de pautas o guías. Por ejemplo, en [Talbi 02], se presenta una de ellas, además de un amplio número de hibridaciones clasificadas con esta taxonomía. El artículo presentado en [Raidl 06] continúa con la idea anterior, combinando el punto de vista expuesto en [Cotta-Porras 98] y en [Alba 05] por Blum et al. con la taxonomía de Talbi. La taxonomía más actual encontrada sobre este tema se encuentra en [Jourdan 09], donde se extiende, de nuevo y siguiendo un enfoque distinto, lo aportado en [Talbi 02] de manera que se tienen en cuenta esquemas cooperativos entre métodos exactos y metaheurísticas. Basándonos en las taxonomías anteriores, se escoge la propuesta en [Talbi 02] para realizar esta clasificación, dado que es la más utilizada incluso por los artículos más actuales.

Siguiendo, por tanto, esta taxonomía, podemos dividir las hibridaciones dependiendo del diseño y la implementación llevada a cabo. Para el caso del diseño encontramos hibridaciones:

- ◇ De nivel bajo, dependiendo de si una función de una de las metaheurísticas es sustituida por otra, o alto, si las diferentes metaheurísticas completas están contenidas en la hibridación.

- ◇ De relevo, donde el conjunto de metaheurísticas se aplican una detrás de otra, o de trabajo en equipo, donde se aplican agentes de cooperación paralela.

- ◇ Homogénea, donde todos los algoritmos utilizan la misma metaheurística, o heterogénea, donde todas las metaheurísticas utilizadas son diferentes.

- ◇ Global, si todos los algoritmos usan el espacio de búsqueda completo, o parcial, si cada uno de los algoritmos utiliza su propio espacio de búsqueda.
- ◇ Especialista, que combina algoritmos que resuelven diferentes problemas, o general, donde todos los algoritmos intentan resolver el mismo problema de optimización.

Para la parte de implementación, la hibridación puede ser:

- ◇ Específica, si solo resuelve un conjunto pequeño de problemas, o de propósito general.
- ◇ Secuencial, donde los algoritmos trabajan uno por uno, o paralela, donde cada algoritmo está trabajando al mismo tiempo que el resto.

Por otra parte, si las metaheurísticas usadas trabajan de manera paralela, se encuentran en una de las siguientes tres categorías:

1. Estáticas, si el número de tareas y la localización del trabajo se generan durante el tiempo de compilación.
2. Dinámicas, si el número de tareas se determina durante la compilación pero la localización del trabajo se determina y/o cambia durante la ejecución.
3. Adaptativas, donde las tareas pueden ser creadas o borradas como una función.

Centrándonos en las características del diseño, el algoritmo propuesto recae en las siguientes categorías:

- ◇ Nivel Alto: la propuesta se encuentra dentro de las hibridaciones de nivel alto, ya que tanto GA como CE están contenidos en su totalidad en la hibridación y se mantienen todos los operadores de cada uno de los métodos hibridados.
- ◇ Trabajo en equipo: durante la ejecución del método propuesto, ambas partes trabajan en paralelo, uniendo los resultados de su ejecución al final de la generación en la creación de la nueva población.

- ◇ Heterogéneo: Si tomamos CE por su naturaleza estadística, la hibridación es heterogénea, ya que GA es una técnica basada en población, siendo, por tanto, ambas diferentes. En cambio, si pensamos en CE como un algoritmo poblacional, ya que mantiene una serie de individuos a lo largo de las generaciones, entonces estaríamos ante un caso de hibridación homogénea.

- ◇ Parcial: Los algoritmos que forman la hibridación son aplicados a dos subpoblaciones diferentes. A su vez, estas sub-poblaciones se forman de manera distinta: La sub-población con la que trabaja el GA se forma a partir del operador de selección mientras que con la que trabaja CE está formado por los mejores individuos de la población actual.

- ◇ General: El algoritmo se aplica a un problema dado con una función objetivo a optimizar, por lo que ambas partes funcionan de manera que se mejore los resultados obtenidos para dicha función, aplicándose, por tanto, al mismo problema objetivo.

Por otro lado, y teniendo en cuenta la parte de implementación, la hibridación desarrollada formaría parte de las siguientes categorías:

- ◇ Propósito general. Aunque sí que es cierto que en este trabajo sólo se aplica la técnica desarrollada a ámbitos específicos como la predicción y la optimización de funciones, el algoritmo está creado de manera que pueda adaptarse a cualquier clase de problema. Bastaría con adecuar la inicialización y los métodos de cruce y mutación del GA, y el cálculo de la media y desviación para la actualización de los individuos en CE, al problema con el que se quiera tratar, además de, obviamente, la codificación de los individuos de la población de la que dependen ambas técnicas.

- ◇ Paralela. La aplicación de las diferentes partes de la técnica se realiza de manera separada gracias a las dos subpoblaciones que se mantienen, reduciendo así el tiempo de cómputo.

El objetivo tras la hibridación que se realiza en este trabajo, como ya se ha mencionado varias veces, es la aportación que ambas técnicas pueden hacer a sus

diferentes ejecuciones. Por lo tanto, lo que se busca es el buen rendimiento de ambos algoritmos en problemas de optimización de índole totalmente diferentes. Será básico, como se hablará en secciones posteriores, encontrar los parámetros correctos para su aplicación en cada uno de los diferentes problemas.

3.3 Sistemas Basados en Reglas Difusas: Aplicación

Anteriormente se ha hablado brevemente de los Sistemas Basados en Reglas Difusas y algunas de sus aplicaciones en problemas de la vida real. En la parte principal del trabajo desarrollado en esta tesis, se han utilizado este tipo de sistemas como herramienta para la predicción de congestión a corto plazo en diferentes puntos de una carretera.

En esta sección, se amplia la información de esta clase de sistemas. Concretamente, de una de sus variantes: el FRBS jerárquico (Hierarchical FRBS, HFRBS) [Fernández 09]. Esta clase de sistemas cuentan con varios FRBSs, los cuáles están unidos unos a otros de tal forma que la salida de uno de ellos es la entrada de otro.

Dependiendo de la forma en la que estructuren los sistemas [Benitez 13], contamos con tres modelos diferentes de HFRBS:

1. HFRBS en serie: la salida de un FRBS es la entrada del siguiente. También es posible incluir variables externas como una entrada del siguiente sistema.

2. HFRBS en paralelo: La estructura de los sistemas está organizada en capas. Las salidas de la primera capa son las entradas de los sistemas de la segunda capa, y así sucesivamente.

3. HFRBS híbridos: Una combinación de los casos anteriores.

La Figura 3.2 muestra un ejemplo de un sistema de jerarquía en serie, mientras que la Figura 3.3 y la Figura 3.4 muestran sistemas paralelos e híbridos respectivamente.

Por otro lado, gracias a su estructura, son útiles para la mejora del equilibrio entre precisión e interpretabilidad en problemas con un gran número de variables, dando lugar a una posible solución al problema de 'maldición de la dimensionalidad' (curse of dimensionality problem), [Benitez 13].

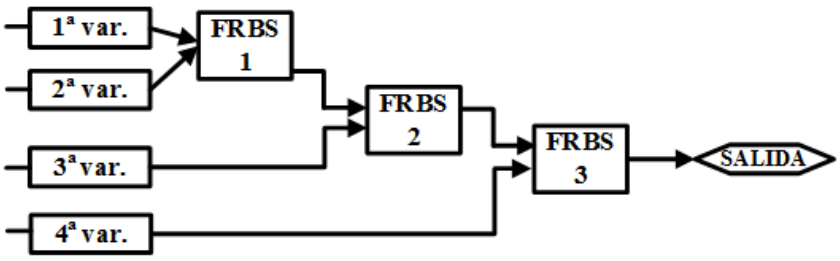

Figura 3.2: Tipos de jerarquías de FRBS: HFRBS en serie.

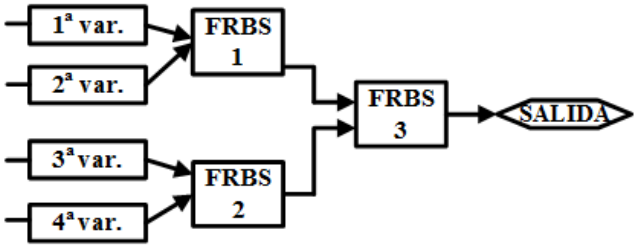

Figura 3.3: Tipos de jerarquías de FRBS: HFRBS en paralelo.

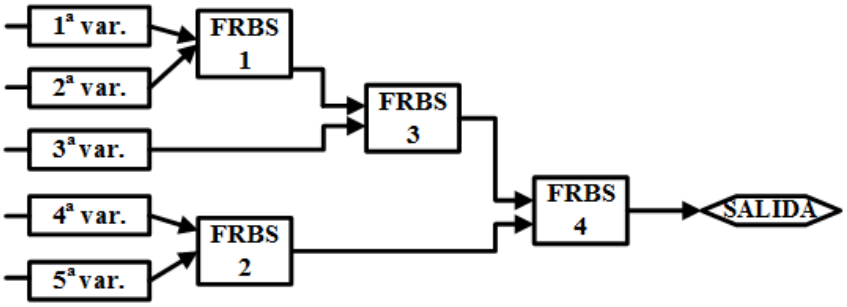

Figura 3.4: Tipos de jerarquías de FRBS: HFRBS híbrido.

Este trabajo está enfocado en el segundo tipo de sistemas: HFRBS en paralelo (*Parallel HFRBS*, PHFRBS). Se ha elegido este tipo de jerarquía porque, de esta manera, todas las variables que pasan al sistema son procesadas al mismo tiempo y con la misma relevancia. Esto se puede observar comparando las Figuras 3.2 y 3.3: mientras que en el primer caso la cuarta variable entra al último sistema, influenciando a la salida sólo en ese momento, en el segundo caso, donde los sistemas están en paralelo, las cuatro variables se tienen en cuenta desde el primer momento, y son las salidas de sus sistemas (FRBS 1 y FRBS 2) las que, usadas como entradas en el último sistema darán lugar a la salida de la jerarquía, teniendo, por tanto, la misma relevancia a la hora de calcular el resultado final.

Utilizamos esta jerarquía con una restricción: cada sistema sólo cuenta con dos entradas. Por lo tanto, si hay un número impar de variables, la última variable será la primera entrada del último sistema. Esto hace que las etiquetas y las reglas utilizadas para cada sistema no se dispare, haciendo imposible en términos de tiempo de cómputo el funcionamiento del sistema, y en términos de representabilidad, su interpretación posterior.

Pongamos como ejemplo de esto lo siguiente: contamos con cuatro variables de entrada y tres etiquetas para cada una de ellas. Con un sistema no jerárquico, contamos con un total de $3^4 = 81$ reglas para cubrir cualquier combinación. En el caso de sistemas jerárquicos como los que nos ocupan, donde cada sistema cuenta con dos entradas, para cuatro entradas, contamos con $Etiquetas^{Entradas} \cdot Sistemas$, donde $Sistemas = Entradas - 1$, es decir, $3^2 \cdot 3 = 27$ reglas para cubrir todas las combinaciones. Generalizando, contamos con un total de $3^N \cdot N - 1$ reglas para N entradas, sabiendo que se necesitan $N - 1$ FRBSs simples para tratarlas, por lo que el número de reglas crece exponencialmente. En el caso de los datasets de congestión, de los que se hablará posteriomente, y que cuentan con un total de 47 entradas, si no se usara jerarquía se necesitarían un total de 3^{47} reglas, mientras que usando jerarquías, contamos con un total de $3^2 \cdot 46 = 414$ reglas.

En el trabajo que nos ocupa, se han utilizado FRBS de tipo TSK, nombrados anteriormente, dado que permite un cálculo rápido de la salida del sistema. Se han utilizado trapecios para la codificación de las entradas y singletons (constantes) para las salidas. Para la inferencia se ha utilizado una T-norma mínimo y agregación por centro de masas (media ponderada).

Año	Artículo	MF	Tipo de Sistema	Estructura	Aprendizaje
1995	[Shimojima 95]	Radial Basis	TSK	Híbrida	GA, Back-propagation
2004	[Chen 04]	Exponencial	TSK	Híbrida	Colonia Hormigas
2007	[Chen 07]	Exponencial	TSK	Híbrida, Serie	PIPE
2008	[Aja-Fernández 08]	Triangular	Mamdani	Paralela, Híbrida	FITM
2009	[Joo 09]	Triangular	TSK	Paralela	Greedy
2010	[Jelleli 10]	Gausiana	TSK	Paralela	Toma de Decisiones
2012	[Benitez 13]	Triangular	Mamdani	Serie	GA Multi-objetivo
2013	[Zhang 14]	Trapecio	TSK	Serie	GA Multi-objetivo
2016	Nuestro trabajo [Lopez-Garcia 16a]	Trapecio	TSK	Paralela	Método Híbrido (GACE)

Tabla 3.1: Tabla de artículos en la literatura de Sistemas Jerarquicos Basados en Reglas.

El algoritmo propuesto realiza en estos sistemas la optimización de sus diferentes partes. De manera breve, dado que en futuras secciones se realizará con más detalle esta explicación, el algoritmo se encarga de optimizar, mediante la hibridización propuesta, las entradas de cada sistema, las etiquetas por las que se regirá la selección de las reglas, y las reglas en sí mismas.

Finalmente, y con el objetivo de reunir una lista de artículos sobre este tipo de sistemas para ampliar el conocimiento sobre estos y su aplicación, la Tabla 3.1 contiene las principales aportaciones de los HFRBS a la literatura en la última década. En dicha tabla se muestra el año de publicación, la referencia al artículo, las funciones de pertenencia (membership functions, MFs) utilizadas, el tipo de sistema, su estructura y el algoritmo de aprendizaje utilizado, detallado en cada uno de los artículos destacados.

3.4 Optimización de funciones: Aplicación

La principal idea tras la aplicación del algoritmo desarrollado en esta tesis a funciones benchmark es demostrar lo siguiente:

- ◇ Su capacidad de adaptación a otro tipo de problemas.
- ◇ La habilidad para alcanzar, sin haberse creado específicamente para esta tarea, un buen resultado en ella.
- ◇ La capacidad del algoritmo de superar, con la combinación de ambas partes, a algoritmos en el estado del arte en esta materia.

- ◇ Comprobar si el cambio de ciertas partes del algoritmo, como pueden ser los métodos de cruce y mutación de la parte genética del mismo, afecta positiva o negativamente a su rendimiento en otros problemas.

Además de cada uno de estos puntos, es crucial para lograr el mayor rendimiento posible, conocer los valores correctos de los parámetros de cada una de las variables que influyen en cada generación del algoritmo. Como se verá en siguientes secciones, este estudio se ha llevado a cabo en la parte de la experimentación dedicada a estas funciones, centrándonos en las probabilidades de cruce y mutación por parte de la parte genética, y del número de individuos que se escogen para actualizar los valores de media y desviación típica que se utilizan en cada una de las ejecuciones de la Entropía Cruzada.

La razón por la que este estudio no se realiza antes de la aplicación del algoritmo a la optimización de los sistemas es simple: en el caso de la optimización de los sistemas, la experimentación está centrada en un ámbito general de estos parámetros, es decir, se toman los establecidos por defecto en la literatura (alta probabilidad de cruce, baja probabilidad de mutación, usar todos los individuos y probabilidad de actualización alta) y se centran los esfuerzos en la cantidad de individuos de cada subpoblación al igual que en la calidad de la solución, número de reglas y etiquetas, y reducción del tiempo de cómputo por la gran cantidad de datos escogidos. Posteriormente, y dada la respuesta positiva en cuanto a rendimiento dada por el algoritmo en este problema, se realiza dicho estudio para confirmar dicho rendimiento en otro tipo de problemas y establecer estos parámetros sin estudios posteriores en otros ámbitos.

3.5 Aspectos a tener en cuenta en la codificación del algoritmo

En este apartado, se recalcan los aspectos a tener en cuenta encontrados durante la codificación y puesta a punto del algoritmo. Dichos aspectos no empeoran la ejecución del mismo, sino que traen en consecuencia tener en cuenta más detalles a la hora de adaptarlo a los diferentes problemas propuestos.

Precisamente, la necesidad de adaptar el algoritmo, concrétamente la inicialización de la población y el cálculo de medias y desviaciones en la parte CE es una de sus limitaciones. Aunque con fácil solución, requiere el estudio previo del problema en cuestión, de manera que la inicialización de los individuos sea la correcta. En cada uno de los problemas que tratamos, la inicialización ha sido diferente, adaptándose al problema en cuestión.

Por otro lado, de entre todos los parámetros a tener en cuenta, el tamaño de la población principal, de la que dependen a su vez el tamaño de las subpoblaciones, merece una mención en esta sección. Aunque se realiza un estudio en capítulos posteriores, es importante conocer que un número elevado de individuos aumenta el tiempo de cómputo del algoritmo a la vez que no mejorará significativamente sus resultados. Esto hace que este tamaño sea pequeño, en torno a 50 individuos, y que ambas poblaciones tengan un máximo de este tamaño como máximo en sus ejecuciones *puras* (sólo GA o sólo CE). Esto deriva en una posible pérdida de diversidad. Al contar con tan poco número de individuos, si el número de generaciones es elevado, o incluso cuando no, todos los individuos de la población terminan siendo el mismo, provocando que, pese a contar con más 'tiempo'para mejorar, no se produzcan cambios. Las soluciones ofrecidas para solucionar esto han sido variadas. Finalizar la ejecución antes de cumplir el criterio de parada si la mejor solución no cambia en un porcentaje de generaciones dado, aumento de la probabilidad de mutación en el caso de que la diversidad disminuya, y el estudio de los parámetros de actualización en el caso del CE son algunas de ellas.

3.6 Implementación del algoritmo propuesto

La motivación de esta sección es mostrar las diferentes implementaciones que se han realizado del algoritmo adaptándose a los problemas propuestos, ampliando lo descrito en la Sección .

Recordando lo descrito en la Sección 3.6, y siguiendo el pseudocódigo descrito en el Algoritmo 3, correspondiente a la primera implementación publicada en [Lopez-Garcia 16a] y aplicada a problemas de predicción de congestión de tráfico, el método propuesto trabaja sobre una población de individuos, denominada Pob_t, que se inicializa de una forma diferente dependiendo de la codificación dada al

problema que nos ocupe (línea 2), aunque siempre de manera aleatoria. Tras dicha inicialización, la población actual se divide en dos subpoblaciones: una población dedicada al GA, llamada GA_{pob}, con un número de individuos GA_{tam}, y una población dedicada al CE, llamada CE_{pob}, con CE_{tam} individuos. Los tamaños de las subpoblaciones son escogidos por el usuario y su suma delimita el tamaño de la población actual. Es decir, $Pob_{tam} = GA_{tam} + CE_{tam}$.

Los individuos de GA_{pob} son elegidos por el operador de selección escogido (línea 7) mientras que los pertenecientes a CE_{pob} son los ejemplos con mejor fitness en Pob_t (línea 8). Una vez formadas ambas subpoblaciones, se trabaja en cada una de manera diferente:

- ◇ En GA_{pob} se aplican los operadores de cruce (línea 9) y mutación (línea 10) para la creación de los nuevos individuos.

- ◇ En CE_{pob} se utiliza un método CE para la generación de nuevos individuos a partir de la media $\overline{x}$ y desviación σ (línea 11). Por otro lado, se escogen un número $Ejemplos_{actualizar}$ de los mejores individuos de esta subpoblación para actualizar los valores $\overline{x}$ y σ de cara a las siguientes generaciones (línea 12).

Con los individuos resultantes de ambas subpoblaciones, nombrados en el Algoritmo 3 como $Hijos_{GA}$ e $Hijos_{CE}$ para el caso de GA_{pob} y CE_{pob} respectivamente, se crea una población completamente nueva Pob_{t+1} que sustituirá completamente a la actual Pob_t (línea 13). Por último, y tras evaluar la población Pob_{t+1}, se aplica elitismo a ésta, añadiendo el mejor individuo encontrado hasta el momento si este no se encuentra en la población (línea 15).

Como se puede comprobar, la mayoría de los parámetros que hay que tener en cuenta para el correcto funcionamiento del algoritmo son dados por el usuario. Tras realizar esta implementación, las preguntas que surgen son las siguientes: ¿Qué parámetros pueden cambiar para mejorar el rendimiento? ¿Cuál es exactamente el número de individuos necesarios para, por ejemplo, actualizar el vector de medias y desviaciones? ¿Qué porcentaje de individuos de la población hay que escoger para cada sub-población?

Datos: $P_{cruce},\ P_{mutacion},\ GA_{tam},\ CE_{tam},\ Learn_{rate},\ Ejemplos_{actualizar}$
Resultado: $Mejor\ individuo\ encontrado$

1 $t \leftarrow 0$
2 $Pob_0 \leftarrow$ Inicializar Poblacion con $GA_{tam} + CE_{tam}$ individuos
3 $\overline{x} \leftarrow$ Inicializar Medias
4 $\sigma \leftarrow$ Inicializar Desviaciones
5 Evaluar Pob_0
6 **mientras** *Condición de parada no alcanzada* **hacer**
7 $GA_{pob} \leftarrow$ Operador de Selección(P_t, GA_{tam})
8 $CE_{pob} \leftarrow$ Seleccionar mejores ejemplos(P_t, CE_{tam})
9 $Hijos_{GA} \leftarrow$ Operador de cruce(GA_{pob}, P_{cruce})
10 $Hijos_{GA} \leftarrow$ Operador de mutación($GA_{pob}, P_{mutacion}$)
11 $Hijos_{CE} \leftarrow$ Generar nuevos individuos($Pob_t, CE_{pob}, \overline{x}, \sigma$)
12 $(\overline{x}, \sigma) \leftarrow$ Actualizar($Learn_{rate}, Ejemplos_{actualizar}, \overline{x}, \sigma$)
13 $Pob_{t+1} \leftarrow Hijos_{GA} \bigcup Hijos_{CE}$
14 Añadir el mejor individuo encontrado a Pob_{t+1} si no estuviera ya
15 Evaluar Pob_{t+1}
16 $t \leftarrow t + 1$
17 **fin**

Algoritmo 3: Pseudocódigo del proceso seguido por GACE cuando se aplica a problemas de predicción de la congestión.

Para responder estas preguntas, se lleva a cabo un estudio posterior a esta experimentación, en el que se intenta, ya no sólo adaptar el tamaño de la población a las diferentes dimensiones del problema, sino saber qué porcentajes del tamaño total de la población deben ser para cada sub-población, y porcentaje de individuos dentro de la población del CE que se debe usar para actualizar sus diferentes vectores. Por tanto, el algoritmo varía, quedando como resultado lo expuesto en el Algoritmo 4, publicado en [Lopez-Garcia 15, Lopez-Garcia 16b].

Los cambios realizados de la primera versión a la que se presenta para la optimización de funciones matemáticas (Algoritmo 4) son los siguientes:

- ◇ Cálculo del tamaño de la población dependiendo del tamaño del problema (línea 2 del Algoritmo 4). En este caso, el tamaño del problema, se multi-

plicará por una constante que determinará el tamaño de la población. En la Sección 4.2 se presenta un estudio orientado a la obtención de este valor.

- ◇ Cálculo del tamaño de la sub-población GA (GA_{tam}) a partir de un porcentaje, y no como un valor fijo, adaptándose así al tamaño de la población Pob_{tam} (línea 2).

- ◇ Por consiguiente, el tamaño de la sub-población CE, CE_{tam} queda calculado como $CE_{tam} = Pob_{tam} - GA_{tam}$ (línea 3).

- ◇ El número de ejemplos utilizados para actualizar la media y desviación, $Ejemplos_{actualizar}$, se calcula ahora a partir de un porcentaje p_{up} dado por el usuario (línea 4).

Los resultados de esta experimentación, así cómo los valores que finalmente se han tomado y con los que se busca ahorrar el cálculo de los mismos en cada problema en el que se aplica el algoritmo, se encuentran en la Sección 4.2, contenida en el siguiente capítulo de esta memoria.

Datos: $P_{cruce},\ P_{mutacion},\ p_{ga},\ Learn_{rate},\ p_{up}$
Resultado: $Mejor\ individuo\ encontrado$
Cálculo de Pob_{tam} a partir del tamaño del problema $GA_{tam} \leftarrow ||Pob_{tam} \cdot p_{ga}||$
$CE_{tam} \leftarrow Pob_{tam} - GA_{tam}$
$Ejemplos_{actualizar} \leftarrow ||CE_{tam} \cdot p_{up}||$
$t \leftarrow 0$
$Pob_0 \leftarrow$ Inicializar Poblacion con $GA_{tam} + CE_{tam}$ individuos
$\overline{x} \leftarrow$ Inicializar Medias
$\sigma \leftarrow$ Inicializar Desviaciones
Evaluar P_0
mientras *Condición de parada no alcanzada* **hacer**
    $GA_{pob} \leftarrow$ Operador de Selección(P_t, GA_{tam})
    $CE_{pob} \leftarrow$ Seleccionar mejores ejemplos(P_t, CE_{tam})
    $Hijos_{GA} \leftarrow$ Operador de cruce(GA_{pob}, P_{cruce})
    $Hijos_{GA} \leftarrow$ Operador de mutación($GA_{pob}, P_{mutacion}$)
    $Hijos_{CE} \leftarrow$ Generar nuevos individuos($CE_{pob}, \overline{x}, \sigma$)
    $(\overline{x}, \sigma) \leftarrow$ Actualizar($Learn_{rate}, Ejemplos_{actualizar}, \overline{x}, \sigma$)
    $Pob_{t+1} \leftarrow Hijos_{GA} \bigcup Hijos_{CE}$
    Evaluar Pob_{t+1}
    Añadir el mejor individuo encontrado a Pob_{t+1} si no estuviera ya
    $t \leftarrow t + 1$
fin

Algoritmo 4: Pseudocódigo del proceso seguido por GACE en la optimización de funciones matemáticas.

Las cosas no se dicen, se hacen,
porque al hacerlas se dicen solas.

Woody Allen

4

Experimentación y resultados

Este capítulo recoge la experimentación realizada con el algoritmo propuesto en esta tesis. La Sección 4.1 cuenta con la explicación de los datasets utilizados, el cálculo de la congestión así cómo la aplicación del método a las diferentes partes de los FRBSs utilizados. Por otro lado, la Sección 4.2 contiene todo lo relacionado con la aplicación de GACE a diferentes funciones matemáticas, el estudio de diferentes parámetros del algoritmo, y los resultados obtenidos por el mismo.

4.1 Aplicación del algoritmo a problemas de predicción de tráfico

A lo largo de esta sección, se encuentra la experimentación llevada a cabo con la técnica propuesta así cómo información sobre los datasets utilizados, operadores y configuraciones de la misma. En la Sección 4.1.1 se encuentra la información sobre los datos utilizados, tal como su obtención, qué información aportan o cómo se van a utilizar en esta experimentación. La codificación de las soluciones se detalla en

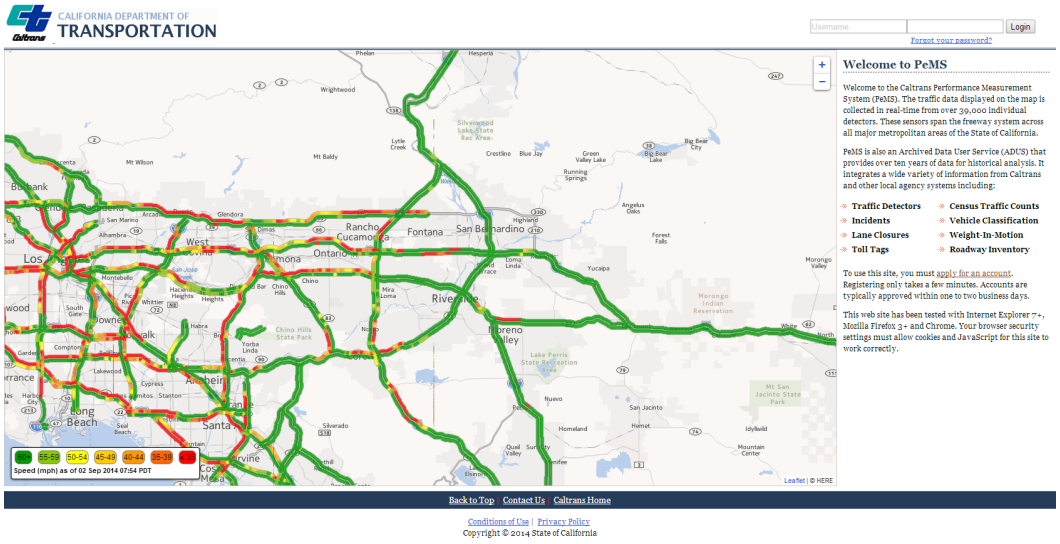

Figura 4.1: Página de Caltrans Performance Measurement System (PeMS)

la Sección 4.1.2 mientras que los operadores utilizados tanto en la parte GA como en la parte CE se encuentran en la Sección 4.1.3. Por último, los resultados de la experimentación se muestran en la Sección 4.1.4

4.1.1 Datos reales de tráfico

Los datos usados en este trabajo son proporcionados por el Caltrans Performance Measurement System (PeMS [1]). En la Figura 4.1 puede verse una captura de pantalla de la página de la plataforma. PeMS es una base de datos en tiempo real del Departamento de Transporte de California que ofrece alrededor de 10 años de datos de tráfico para su análisis. Para este trabajo, se ha usado una sección de la carretera I5 en Sacramento, California, de 5.60 millas (apróx. 9 kilómetros).

Concrétamente, el tramo elegido cuenta con 13 puntos situados en la carretera principal y 8 sensores en las entradas y salidas de la vía, de los cuáles también se han recogido datos. Las medidas de tráfico son recogidas por dichos sensores con una frecuencia de 5 minutos. La información que recogen los sensores de la carretera principal consiste en el flujo (número de vehículos), la ocupación (el porcentaje de tiempo que el sensor está activo) y la velocidad (en millas por hora). La

[1] www.pems.dot.ca.gov

Nivel de congestión	Densidad de Tráfico (ve/mi/ln)	Velocidad del vehículo (mi/h)
Ligera	[46–60]	[30–50]
Moderada	[60–80]	[15–40]
Severa	> 80	< 25
Nula	Resto de casos	

Tabla 4.1: Valores de congestión y su cálculo.

información que se obtiene de los sensores situados en las entradas y salidas sólo concierne a la fluidez. Los datos se han obtenido en el intervalo de tiempo desde las 0:00 horas del 1 de Septiembre de 2013, hasta las 23:55 del 30 de Septiembre del mismo año. La Figura 4.2 muestra un esquema de la carretera así como la localización de los sensores en la misma.

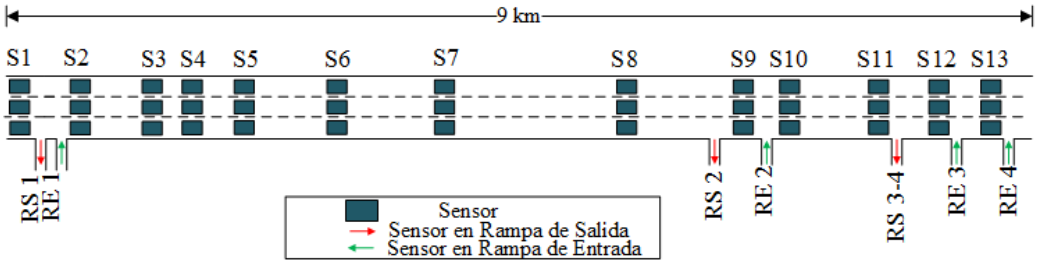

Figura 4.2: Segmento de la carretera I5. Los sensores se denotan por S, las rampas de salida por RS y las de entrada por RE.

Dado que sólo disponemos de información puntual, se utilizan los intervalos propuestos en [Skycomp 09] y mostrados en la Tabla 4.1 para definir el nivel de congestión en un momento dado, y poder usar estos datos para crear modelos que lo predigan. Por otro lado, aunque la densidad de vehículos no es proporcionada por los datos de PeMS, se calcula usando los valores de velocidad y fluidez: $densidad = fluidez/velocidad$. La densidad sólo es utilizada para calcular la congestión y no se incluye como dato en los datasets.

En total, en cada instancia de un dataset completo se almacenan un total de 48 variables cada 5 minutos:

- $\{F_i, O_i, V_i\}$, para todo $i = 1 \ldots 13$, representando el flujo, la ocupación y la velocidad respectivamente en el correspondiente sensor de la vía principal.

- In_j e $Out_j\}$, para todo $j = 1 \ldots 4$, representando el flujo de la correspondiente entrada o salida de la vía.

- $Congestion$, como variable de clase a determinar, que puede tomar los valores $Congestion = \{Nula, Ligera, Moderada, Severa\}$ calculados según la Tabla 4.1.

4.1.1.1 Conjunto de datos

Con los datos de los que se ha hablado en la sección anterior, se han creado cuatro tipos de conjuntos de datos, o datasets:

1. Dataset Punto (DP): este conjunto de datos contiene las medidas de todos los sensores. La congestión se calcula en un único punto de la carretera. En este caso, se ha escogido como dicho punto el nombrado como $S7$.

2. Dataset Punto Simplificado (DPS): este conjunto de datos es una versión simplificada de DP. Sólo contiene los datos del primer ($S1$), séptimo ($S7$) y último ($S13$) sensor y una combinación de los valores de flujo de las rampas de entrada y salida. Este valor es la media de los flujos antes y después del punto de interés. El valor de la congestión se calcula de la misma forma que en DP. La Figura 4.3 muestra la forma en la que se han obtenido estos datos.

3. Dataset Sección (DS): este dataset contiene las medidas de todos los sensores y la congestión se calcula como el máximo nivel de congestión que se da en cualquiera de los sensores de la carretera.

4. Dataset Sección Simplificado (DSS): versión simplificada de DS. Sólo contiene los datos del primer ($S1$), séptimo ($S7$) y último ($S13$) sensor y la media del valor fluidez de las rampas de entrada y salida. La congestión se calcula de la misma forma que en DS.

La Tabla 4.2 contiene una clasificación de los conjuntos de datos utilizados dependiendo de los sensores de los que se han obtenido los datos y el cálculo de la congestión en cada uno de ellos.

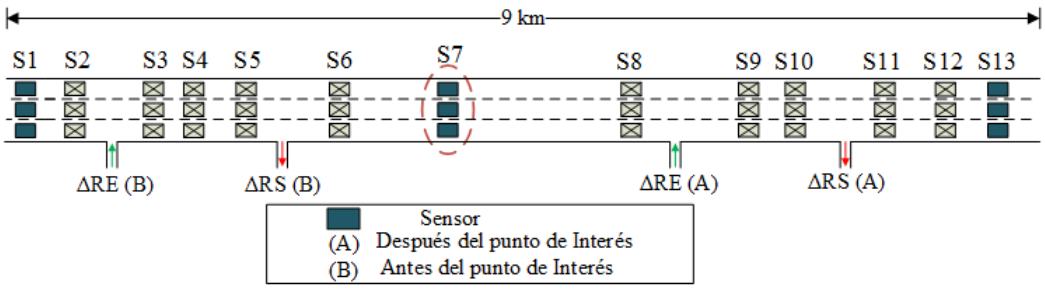

Figura 4.3: Dataset Punto Simplificado. Sólo tiene en cuenta los sensores $S1$, $S7$ y $S13$, y una combinación de las rampas de entrada y salida antes y después del punto de interés $S7$.

Sensores \ Congestión	Máximo Sección	Punto
Todos	DS	DP
$S1$,$S3$,$S7$	DSS	DPS

Tabla 4.2: Clasificación de los datasets según el número de sensores utilizados y el cálculo de la congestión

El objetivo de DP y DPS es predecir la congestión en un único punto $S7$, mientras que DS y DSS pretenden predecir el máximo nivel de congestión que puede darse en la sección de carretera completa. Por otra parte, DS y DP utilizan toda la información disponible (las 47 variables más la clase anteriormente descritas) mientras que los conjuntos simplificados utilizan una versión con un menor número de atributos, 13 atributos más la variable de clase. Estos atributos son la fluidez, ocupación y velocidad en los sensores antes descritos (9 variables) y cuatro valores de flujo en las entradas y salidas antes y después del punto de interés, más la variable de clase.

Además, por cada tipo de dataset, para realizar la predicción de la congestión, la salida deseada será un valor de congestión a ocurrir en un horizonte de tiempo de $\{5, 10, 30\}$ minutos. Por lo tanto, se utilizarán un total de 12 conjuntos de datos que serán denotados con su acrónimo y el horizonte de tiempo. Por ejemplo, DP_5 será el correspondiente al Dataset Punto con un horizonte de 5 minutos. Otro ejemplo podría ser DSS_{15}, que correspondería al Dataset Sector Simplificado con

Datasets	Nº Var	Nula	Ligera	Moderada	Severa	IR
DP_5	47	8277 → 850	61→ 61	172 → 172	129 → 129	135.6→ 13.9
DP_{15}	47	8275 → 866	61 → 61	172→ 172	129→ 129	135.6→ 14.1
DP_{30}	47	8272→ 847	61→ 61	172 → 172	129→ 129	135.6→ 13.8
DPS_5	13	8277→ 113	61→ 55	172 → 120	129→ 97	135.6→ 2.1
DPS_{15}	13	8275→ 139	61 → 45	172 → 102	129→ 103	135.6→ 3.0
DPS_{30}	13	8272→ 88	61 → 58	172 → 108	129 → 96	135.6→ 1.8

Tabla 4.3: Datasets Punto y Punto Simplificado. Los valores antes de la fecha indican su valor original, mientras que los que aparecen después se obtienen tras la reducción.

un horizonte de predicción de 15 minutos.

Los datasets utilizados en este trabajo contienen un número muy alto de ejemplos libres de congestión con respecto a ejemplos con otros tipos de congestión. Por esta razón, se puede decir que los conjuntos de datos utilizados están altamente no balanceados. Para dar validez a dicha afirmación, se utiliza el llamado 'Imbalance Ratio' (IR) [López 13]. La Ecuación 4.1 contiene el cálculo de dicho ratio. Además, la Tabla 4.3 muestra el número de instancias de cada tipo de congestión incluidas en cada uno de los datasets. Estos valores, y otros como el número de variables en cada conjunto y el número de instancias de cada clase se muestran en la Tabla 4.3.

$$IR = \frac{MAC}{MIC} \tag{4.1}$$

donde MAC es el número de instancias de la clase mayoritaria y MIC el número de instancias de la clase minoritaria.

Con el objetivo de balancear los conjuntos de datos, se ha llevado a cabo un proceso de simplificación basado en el método de los K-Vecinos [Keller 85]. La reducción se basa en dos parámetros: k y u, donde k indica el número de vecinos escogido y u el umbral que no se puede superar por la distancia entre el nodo actual y cualquiera de los vecinos escogidos. Si este umbral es superado, se elimina el enésimo vecino del dataset. El método es iterativo y se detiene cuando no hay nodos que puedan ser eliminados. Es decir, cuando ninguna de las combinaciones entre el nodo y los vecinos supere el umbral establecido. Los resultados obtenidos

Datasets	Nº Var	Nula	Ligera	Moderada	Severa	IR
DS_5	47	4157→ 293	2776 → 473	1402 → 453	304 → 304	13.6→ 1.6
DS_{15}	47	4155 → 290	2776 → 549	1402 → 444	304 → 304	13.6→ 1.8
DS_{30}	47	4152→ 293	2776 → 464	1402 → 442	304→ 304	13.6→ 1.5
DSS_5	11	4157→ 46	2776 → 58	1402 → 65	304 → 108	13.6→ 2.3
DSS_{15}	11	4155→ 56	2776 → 57	1402 → 85	304 → 110	13.6→ 1.9
DSS_{30}	11	4152 → 54	2776 → 43	1402 → 76	304 → 114	13.6→ 2.6

Tabla 4.4: Datasets Sector y Sector Simplificado. Los valores antes de la fecha indican su valor original, mientras que los que aparecen después se obtienen tras la reducción.

aplicando esta técnica a los datasets se muestra en la Tabla 4.3 después de cada flecha. Esta tabla también contiene el valor IR de cada conjunto para que pueda ser comparado con los datasets completos. Como se puede observar en dicha tabla, la clase mayoritaria siempre es Nula y la minoritaria Severa o Ligera antes de la simplificación. Tras ella, varía entre los distintos valores que puede tomar la congestión, llegando incluso a ser la clase mayoritaria Severa en los datasets DSS. En el caso del valor de IR, es mucho mayor en los datasets de Punto que en los de Sección. Posteriormente a la reducción de instancias, el IR más alto corresponde a los datasets DP, mientras que, para el resto, ronda los mismo valores (entre 1.6 y 3.0).

Los datasets reducidos se usan como conjuntos de entrenamiento para el algoritmo, con la idea de evitar sobreentrenar las clases mayoritarias, mientras que los datasets completos se usan para testear los resultados.

4.1.2 Codificación de las soluciones

Para poder explicar varios aspectos del algoritmo y calcular diferentes valores del cromosoma, se introduce en esta sección su estructura. El cromosoma codifica las tres partes que se utilizarán en cada uno de los FRBS que componen un PHFRBS:

1. Jerarquía: se define como el subconjunto de variables seleccionados para ser procesada por el PHFRBS, así como el orden en el que estas deben ser incluidas en el sistema.

2. Funciones de pertenencia (MF): contiene la localización de las MFs utilizadas para codificar cada una de las variables de cada sistema FRBS en la jerarquía.

3. Base de reglas (Rule Base, RB): codifica las posiciones de los singletons usados como consecuentes de la base de reglas de un sistema FRBS en la jerarquía.

A lo largo de esta tesis, llamaremos a estas partes $C_{jerarquia}$, $C_{etiquetas}$ y C_{reglas} respectivamente. Cada una se describe con detalle a continuación.

$C_{jerarquia}$ es una permutación en la cual un valor i en la posición j denota que la i-ésima variable se inserta en el PHFRBS en la j-ésima posición. Además, un carácter de terminación, denotado como 0, determina a partir de qué punto del vector no se utilizan más variables. Por lo tanto, el tamaño de la permutación es de $N_{variables} + 1$. El número de módulos o sistemas ($N_{modulos}$) está relacionado con $N_{variables}$: el valor máximo que puede tomar $N_{modulos}$ es $N_{variables} - 1$. Las Figuras 3.3 y 3.4 ilustran esta afirmación.

$C_{etiquetas}$ está compuesta por dos matrices de valores reales en el intervalo [-1, 1], llamadas MF_1 y MF_2 , que contienen la localización de las funciones de pertenencia de la primera y segunda variable de cada uno de los sistemas individuales FRBS respectivamente. De manera formal, $C_{etiquetas}$ está formado por dos matrices de $N_{modulos} \cdot N_{variables}$ elementos. De una manera formal contamos con:

$$C_{etiquetas} = MF_i(j,k)\ \forall i \in \{1,2\},\ \forall j \in \{1 \dots N_{modulos}\},\ \forall k \in \{1 \dots N_{etiquetas}\} \tag{4.2}$$

donde cada $MF_i(j,k)$ denota la k-ésima función de pertenencia MF usada para codificar la i-ésima entrada del j-ésimo sistema FRBS en la jerarquía. Además, se usa una técnica denominada 'lateral tuning' presentada en [Alcalá 07] para la codificación de las MF. Originalmente, los valores contenidos en las MF están normalizados para ajustarse al intervalo [-1, 1]. Primero, se calculan las divisiones necesarias basadas en el número de etiquetas que usa el problema. Posteriormente, el método calcula la posición de las MFs en estas divisiones y convierte su valor en un intervalo $[min, max]$.

La Figura 4.4 muestra cómo trabaja esta codificación para una estructura de cuatro etiquetas. En gris aparecen las etiquetas si estuvieran uniformemente distribuidas. Podemos comprobar como, con $X_3 = 0.0$, el punto coincide con la etiqueta uniforme, mientras que, con valores negativos, como en $X_2 = -0.2$, el valor se encuentra ligeramente a la izquierda de la etiqueta correspondiente.

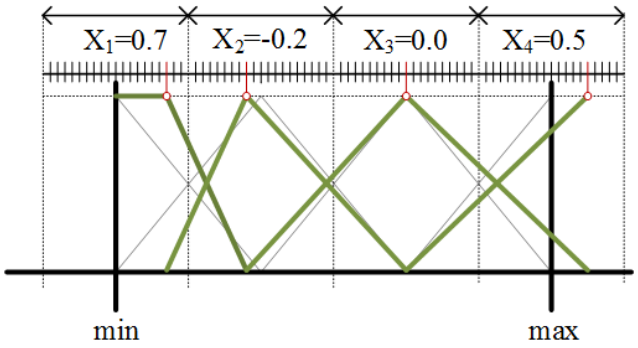

Figura 4.4: Ejemplo de codificación 'lateral tuning'para cuatro MFs. Cada X_i puede tomar un valor en el intervalo [-1, 1].

Finalmente, se presenta C_{reglas} como una matriz de valores reales en el intervalo [0, 1]. Por cada sistema individual FRBS, su base de reglas RB tiene un tamaño de $N^2_{etiquetas}$. Hay que tener en cuenta que contamos con bases de reglas completas. Es decir, para cada entrada, y cada etiqueta de dicha entrada, contamos con una salida. Así pues, con un sistema i de dos entradas y tres etiquetas por entrada, las reglas, con j definida en $j \in \{1 \dots 9\}$, serían:

$$Si\ Entrada_1 = ET1_1\ y\ Entrada_2 = ET2_1\ entonces\ Regla_{(i)(1)}$$

$$Si\ Entrada_1 = ET1_1\ y\ Entrada_2 = ET2_2\ entonces\ Regla_{(i)(2)}$$

$$\dots$$

$$Si\ Entrada_1 = ET1_3\ y\ Entrada_2 = ET2_2\ entonces\ Regla_{(i)(j-1)}$$

$$Si\ Entrada_1 = ET1_3\ y\ Entrada_2 = ET2_3\ entonces\ Regla_{(i)(j)}$$

De una manera más formal, las reglas estarían definidas como sigue:

$$C_{reglas} = Reglas_i(j)\ \forall i \in \{1 \dots N_{modulos}\},\ j \in \{1 \dots N^2_{etiquetas}\} \tag{4.3}$$

donde cada valor denota el consecuente de la regla j-ésima de la base de reglas del i-ésimo sistema FRBS en la jerarquía.

Como ejemplo, la Figura 4.5 presenta una codificación de un PHFRBS con seis variables. Como se muestra en dicho ejemplo, $C_{jerarquia}$ determina el orden en el que las variables entrarán a la jerarquía, así como las variables que serán excluidas (mediante el uso del carácter de terminación). Por otro lado, las funciones de pertenencia MF y la base de reglas RB del i-ésimo FRBS en la jerarquía están denotadas como $MF_1(i, m)$, $MF_2(i, m)$ y $Reglas_i(r)$, donde $m \in \{1, \dots N_{etiquetas}\}$ y $r \in \{1 \dots N^2_{etiquetas}\}$.

Es importante recalcar que tanto $C_{etiquetas}$ y C_{reglas} están limitadas respectivamente a los intervalos [-1, 1] y [0, 1]. Por lo tanto, todos los operadores implementados deben tomar los valores dentro de los correspondientes intervalos.

Hay que tener en cuenta que para la optimización de un PHFRBS se han utilizado dos codificaciones diferentes en el mismo individuo: la permutación de $C_{jerarquia}$ y los valores reales tanto de $C_{etiquetas}$ como de C_{reglas}. Por esta razón, se han usado operadores específicos que trabajan con estas codificaciones.

4.1.3 Operadores

En esta sección, se desarrollan los operadores utilizados en la parte genética del algoritmo propuesto (Sección 4.1.3.1) y los operadores utilizados en la parte de la Entropía Cruzada (Sección 4.1.3.2).

4.1.3.1 Operadores utilizados en el Algoritmo Genético

En este apartado se expone la explicación de los diferentes operadores usados en la parte del GA. Es decir, la definición de los operadores de selección, cruce y mutación utilizados en esta experimentación se encuentra en esta sección.

Se usa como operador de selección el Torneo Binario [Goldberg 91]. El operador escoge dos individuos aleatoriamente dentro de la población. El ganador del

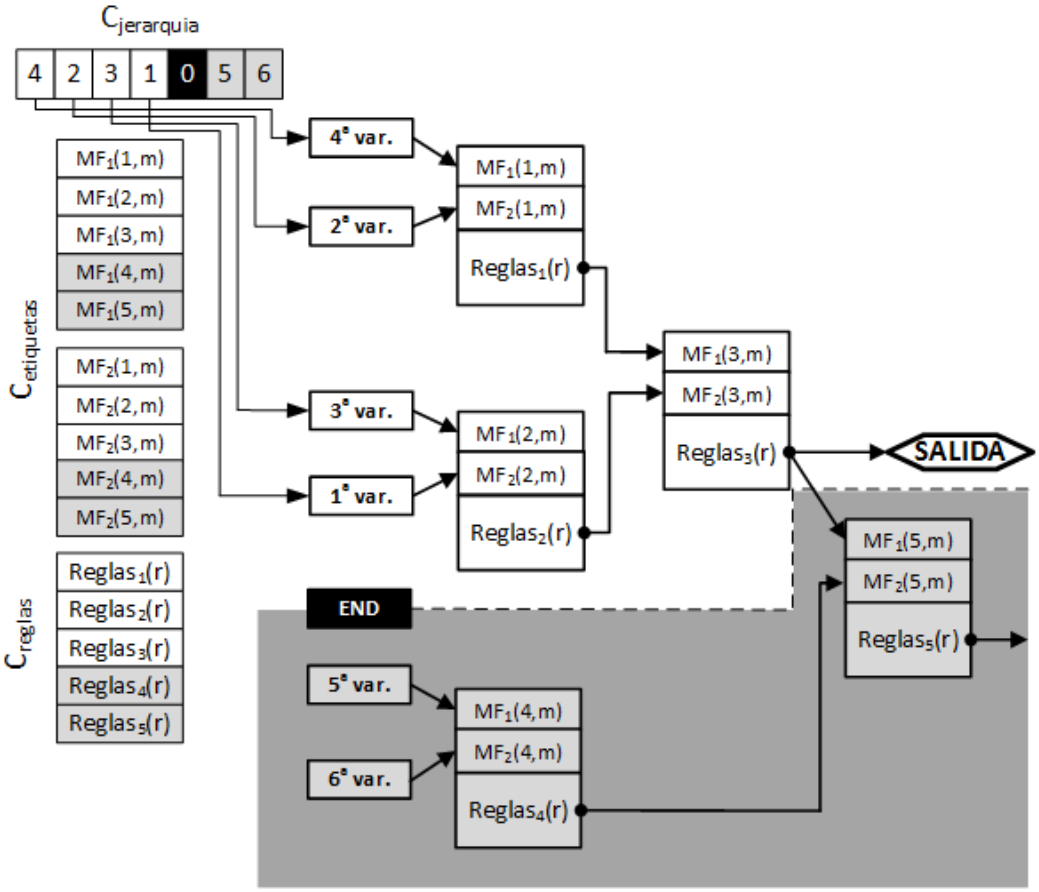

Figura 4.5: Ejemplo de codificación de un PHFRBS con seis variables de entrada.

torneo es el individuo con mejor fitness. El proceso se repite, en el caso que nos ocupa, hasta haber escogido GA_{tam} individuos.

Aunque las etiquetas y las reglas de nuestros individuos son matrices que contienen valores reales, la jerarquía es una permutación, como se verá en la Sección 4.1.2. Por lo tanto, son necesarios dos operadores de cruce y otros dos de mutación.

Para la permutación, se escoge una variante del Order Crossover [Deep 10], donde sólo se selecciona un punto para realizar la operación. El operador escoge un punto c de manera aleatoria y conserva la permutación de los padres desde la primera posición hasta el punto elegido.

El resto de la jerarquía del hijo se completa de acuerdo a los valores del padre cuyo primer segmento no se ha heredado. La decisión de usar un único punto viene

dada por querer mantener las primeras variables a lo largo de la herencia en su posición dado que son las que más probabilidades tienen de entrar en el PHFRBS (siempre que estén delante del carácter de terminación). La Figura 4.6 muestra un ejemplo. Considerando dos padres (P_1 y P_2), los hijos estarían formados en primera instancia por $\{2, 3, 4\}$ y $\{4, 5, 2\}$ (cajas claras). Después, empezando por el punto de corte c, los valores son escogidos en el orden dado por el otro padre, obviando los valores que ya se encuentran en los hijos, marcados con una x. Por tanto, la secuencia para O_1 es $\{1, 6, 5\}$ y para O_2 es $\{6, 1, 3\}$. Esta secuencia se coloca posteriormente al punto de corte en el hijo correspondiente.

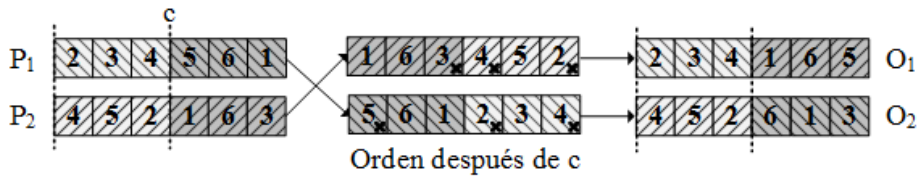

Figura 4.6: Variante del cruce de orden usada en el trabajo. Primero, se elige el punto de corte. Posteriormente, se selecciona el orden y finalmente se crean los nuevos individuos.

Para la parte real de los individuos (etiquetas y reglas), se ha escogido el método de cruce BLX-α [Herrera 98, Eshelman 93]. Dado dos padres $A = (a_1 \dots a_i)$ y $B = (b_1 \dots b_i)$ por cada i, BLX-α crea dos hijos generando valores aleatorios en el intervalo presentado en la Ecuación 4.4 con $\alpha \in$ [0,1].

$$[min(a_i, b_i) - \alpha \cdot |a_i - b_i|,\ max(a_i, b_i) + \alpha \cdot |a_i - b_i|] \tag{4.4}$$

Para los operadores de mutación, también se realiza una distinción entre jerarquía, etiquetas y reglas. Para la jerarquía, se aplica un operador de intercambio [Larranaga 99]: Se escogen dos posiciones dentro de la permutación y estas son cambiadas la una por la otra. Para las etiquetas y las reglas, se aplica una mutación BGA [Mühlenbein 93]. Dado $A = (a_1 \dots a_m)$, el operador devuelve a_i, calculado tal y como se presenta en la Ecuación 4.5

$$a_i' = a_i \pm \beta \cdot \sum_{k=0}^{15} (\alpha_k 2^{-k}) \tag{4.5}$$

Individuo	$\overline{x_0}$	σ_0
$C_{jerarquia}$	$0.5 \cdot N_{variables}$	$0.5 \cdot N_{variables}$
$C_{etiquetas}$	0	1
C_{reglas}	0.5	0.5

Tabla 4.5: Valores iniciales por cada una de las partes de $\overline{x}$ and σ.

donde β define el rango de mutación. El signo (+ or -) se escoge con probabilidad 0.5, $\alpha_k \in \{0, 0.33, 0.66, 1\}$ se genera aleatoriamente con la siguiente probabilidad:

$$p(\alpha_k = \{0, 0.33, 0.66, 1\}) = \frac{1}{4}. \tag{4.6}$$

4.1.3.2 **Operadores utilizados en la Entropía Cruzada**

Como se dijo antes, deben mantenerse dos individuos a lo largo del CE: uno que mantenga la media $\bar{x}$ y otro que guarde la varianza σ. Estos individuos tienen la misma estructura que un individuo normal (Figura 4.5) pero se inicializan de tal manera que generar nuevos ejemplos a partir de ellos (en la primera iteración) sea equivalente a generar individuos aleatorios. Para dicho propósito, la inicialización de cada una de sus partes se realiza tal y como se presenta en la Tabla 4.5. Como valor medio, se utiliza el punto medio del intervalo en el que se pueden dar los valores de cada una de las partes del individuo mientras que como desviación se toma la mitad de la amplitud de dicho intervalo. Estos valores iniciales se actualizarán durante la ejecución del algoritmo.

Los individuos que procesa el CE son seleccionados de manera determinística. Los CE_{tam} mejores individuos de la población constituyen la población CE_{pob}. Una vez que se obtengan, los individuos $\bar{x}$ y σ actualizan sus valores y se generan los nuevos ejemplos.

La actualización se realiza aplicando directamente las ecuaciones de las líneas 7 y 8 del Algoritmo 2 a las etiquetas ($C_{etiquetas}$) y reglas (C_{reglas}) de los individuos. Dichas partes en los nuevos individuos se generan siguiendo una distribución normal con $\bar{x}$ y σ.

Para la permutación ($C_{jerarquia}$) de los individuos, tanto la actualización de los valores media y varianza como la generación de los nuevos ejemplos sigue un pro-

ceso diferente. Los vectores que representan la jerarquía se convierten en vectores que guardan el orden de cada una de las variables. En el vector de orden, el valor almacenado en la posición i representa la posición que ocupa el valor i en el vector de permutación. La última posición se usa para representar la posición del carácter de terminación. Entonces, $\bar{x}$ y σ se actualizan usando estos vectores de orden. Finalmente, se expone a los ejemplos generados a la transformación inversa: de vectores de orden a vectores de jerarquía.

Este proceso queda patente en la Figura 4.7 y funciona tal que: dados los individuos seleccionados por el CE (a), se convierten a vectores de orden en (b) escogiendo la posición en la que se encuentra cada variable. Posteriormente, se obtienen los valores media y varianza de dichas posiciones y se actualiza el valor de $\bar{x}$ y σ (c). Por último, se generan los nuevos ejemplos a partir de dicha media y varianza en (d) y se realiza el proceso inverso de transformación de vectores de orden a vectores de jerarquía (e).

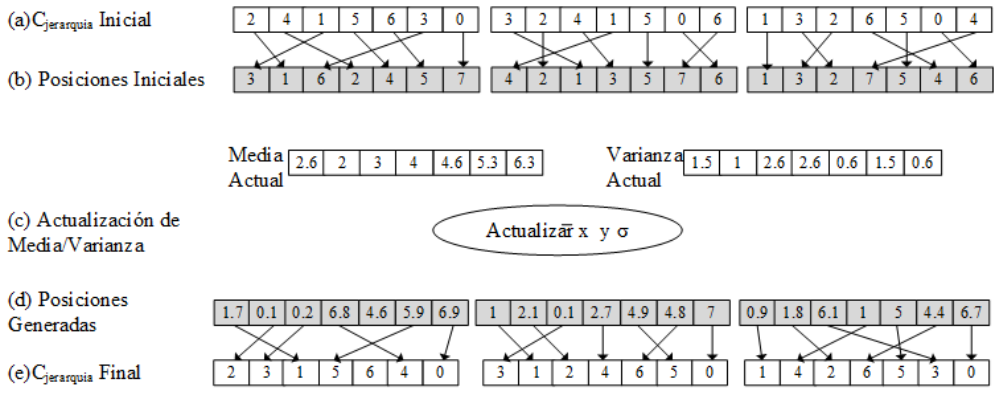

Figura 4.7: Fases de creación de la parte $C_{jerarquia}$ en un nuevo individuo.

Para explicar de manera más exhaustiva el paso de vector de jerarquía a vector de orden, se crea la Figura 4.8. Contando con un vector de jerarquía J_1, cuyas variables se encuentran de la siguiente manera $J_1 = \{2, 4, 1, 3, 0\}$, éstas pasarían a ocupar en el vector de orden O las posiciones que ocupan en el vector J_1, es decir, la variable 1 se encuentra en la tercera posición, la variable 2 se encuentra en la primera, y así sucesivamente. Por tanto, el vector O queda definido tal que $O = \{3, 1, 4, 2, 5\}$. Para volver a crear el vector de jerarquía a partir de un

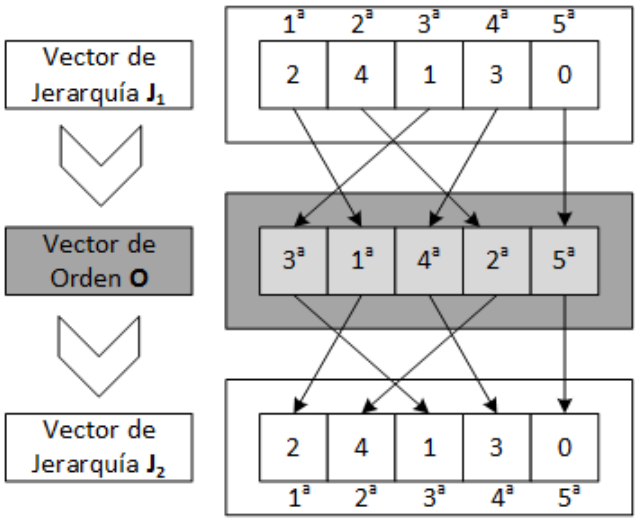

Figura 4.8: Transformación de vector de jerarquía a vector de orden y viceversa

vector de orden, se realiza el paso contrario. Sea $O = \{3, 1, 4, 2, 5\}$, las variables se encontrarán en la posición que ocupan en el vector O, es decir, la variable 1 se encontrará en la posición 3 del vector de jerarquía, la variable 2 en la posición 1 y así sucesivamente. Por tanto, el vector de jerarquía J_2 será igual a $J_2 = \{2, 4, 1, 3, 0\}$. Cabe destacar que la variable de terminación establecida como 0 no tiene porqué ocupar siempre el último lugar.

Los individuos generados se combinan con los que se han creado en la parte de GA, generando la población que se procesará en la siguiente iteración del algoritmo.

4.1.4 Configuración y resultados

Se escogen combinaciones diferentes del número de individuos en GA_{pob} y CE_{pob} para realizar los experimentos y comparar sus resultados. Además, las configuraciones utilizadas en el GACE se comparan con un GA puro ($CE_{tam} = 0$) y con un CE puro ($GA_{tam} = 0$) para comprobar los beneficios de la hibridación realizada frente a la aplicación de los métodos por separado.

Se llevan a cabo ocho experimentos con diferente número de individuos en la población. Cada ejecución se repite 10 veces para obtener los resultados medios.

Para los experimentos, se ha establecido el número de generaciones a 500 y el tamaño total de la población a 50. Las ejecuciones se nombran por el tamaño de sus poblaciones, es decir, GA_{tam}–CE_{tam}. Por ejemplo, $GACE_{50-0}$ se refiere a un GA con los operadores explicados en esta memoria, y $GACE_{0-50}$ corresponde a una ejecución de una CE. El tamaño de la población del GA se encuentra en $GA_{tam} \in \{50, 45, 40, 35, 25, 15, 10, 0\}$ mientras que $CE_{tam} = 50$–GA_{tam}.

El número de funciones de pertenencia usado en cada uno de los sistemas FRBS que componen la jerarquía está establecido a $N_{etiquetas} = 3$, por lo que el número de reglas en cada sistema individual es igual a $N_{reglas} = N^2_{etiquetas} = 9$. El tamaño de la permutación depende del número de variables de cada conjunto de datos más uno (el carácter de terminación), y varía entre 12 y 48 (Tablas 4.3 y 4.4).

Para la parte de la hibridación correspondiente al GA, la probabilidad de cruce está establecida a 0.8 y la de mutación a 0.2. Tanto para el cruce BLX como la mutación BGA, $\alpha = \beta = 0.5$. En la parte del CE, el $Learn_{rate}$ se ha establecido a 0.7 para actualizar tanto $\bar{x}$ como σ.

Para medir el error se aplica en primera instancia el llamado Error Medio Absoluto (Mean Absolute Error, MAE). Las Tablas 4.6 y 4.7 presentan el porcentaje medio de error en los conjuntos de test así como su desviación típica. Los resultados resaltados indican los dos mejores valores para cada dataset.

Se puede decir, observando dicha tabla, que en la mayoría de los casos, un GACE con $GA_{tam} \in [35, 45]$ obtiene la mejor precisión frente a otros algoritmos como $GACE_{25-25}$, GA o CE. Por lo tanto, podemos afirmar que se obtiene un rendimiento mayor por los algoritmos que usan un mayor GA_{tam} que CE_{tam}. Analizándolo a fondo, $GACE_{40-10}$ obtiene uno de los dos mejores resultados en 7 de los 12 casos mientras que $GACE_{45-5}$ lo hace en 8 de 12 casos. Por otra parte, tanto $GACE_{10-40}$ como CE no se encuentran nunca entre los mejores resultados. Comparando GA con CE, GA obtiene el mejor resultado en todos los casos mientras que CE obtiene el peor resultado en 9 de los 12 casos de estudio.

Por otro lado, MAE no tiene en cuenta qué separadas están entre sí la predicción realizada frente a la esperada. Es decir, este error se calcula teniendo presente que la instancia predicha no es igual a la esperada, pero no cómo difiere el tipo de congestión predicho del esperado. Un ejemplo: no es lo mismo predecir una congestión Nula y que se espere una de tipo Leve a predecir una congestión Nula y

Dataset		GA	GACE $45-5$	GACE $40-10$	GACE $35-15$	GACE $25-25$	GACE $15-35$	GACE $10-40$	CE
DP_5	$\overline{MAE}$	0.014	**0.009**	**0.009**	0.010	0.012	0.017	0.017	0.040
	MAE_σ	0.002	**0.001**	**0.002**	0.001	0.002	0.007	0.002	0.034
DP_{15}	$\overline{MAE}$	0.027	**0.013**	**0.013**	**0.013**	0.015	**0.014**	0.019	0.028
	MAE_σ	0.039	**0.002**	**0.001**	**0.001**	0.001	**0.001**	0.006	0.009
DP_{30}	$\overline{MAE}$	0.018	0.017	**0.016**	**0.016**	0.018	0.029	0.021	0.027
	MAE_σ	0.001	0.001	**0.001**	**0.001**	0.002	0.028	0.003	0.011
DPS_5	$\overline{MAE}$	0.012	0.011	**0.008**	**0.009**	0.010	0.033	0.058	0.220
	MAE_σ	0.006	0.003	**0.001**	**0.002**	0.002	0.043	0.097	0.133
DPS_{15}	$\overline{MAE}$	0.021	0.019	**0.015**	**0.015**	0.018	0.019	0.025	0.164
	MAE_σ	0.015	0.007	**0.003**	**0.002**	0.005	0.004	0.018	0.144
DPS_{30}	$\overline{MAE}$	**0.015**	**0.015**	**0.015**	0.018	**0.014**	0.017	0.020	0.158
	MAE_σ	**0.002**	**0.002**	**0.001**	0.011	**0.001**	0.004	0.006	0.129

Tabla 4.6: Media de MAE y desviación en los conjuntos de prueba de cada una de las técnicas para datasets Punto y Punto Simplificado

que la esperada sea Severa. En este ejemplo, la segunda instancia debería obtener un mayor error que la primera.

Teniendo esto en cuenta, también se ha aplicado en segunda instancia otro tipo de error llamado sMAPE (Symmetric Mean Absolute Percentage Error) [Hyndman 06]. El cálculo de este error se muestra en la Ecuación 4.7, donde $\overline{Y}$ es el valor esperado, Y el valor predicho, y n el número de ejemplos. Usando este error, solucionamos el problema anterior.

$$sMAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\overline{Y_i} - Y_i|}{(|\overline{Y_i}| + |Y_i|)/2} \tag{4.7}$$

Los mejores resultados por cada conjunto de datos se encuentran resaltados en las Tablas 4.8y 4.9. Como se ve por dichas tablas, lo obtenido anteriormente queda también constatado en estos resultados: GACE con $GA_{size} \in [35,\ 45]$ obtienen mejor resultado que el resto de configuraciones. $GACE_{45-15}$ obtiene esta vez uno de los dos mejores resultados en 9 de 12 casos de estudio, mientras que $GACE_{40-10}$

Dataset		GA	GACE $45-5$	GACE $40-10$	GACE $35-15$	GACE $25-25$	GACE $15-35$	GACE $10-40$	CE
DS_5	$\overline{MAE}$	**0.174**	**0.149**	0.184	0.195	0.175	0.206	0.212	0.200
	MAE_σ	**0.066**	**0.062**	0.047	0.053	0.036	0.017	0.026	0.030
DS_{15}	$\overline{MAE}$	0.176	**0.130**	0.182	**0.169**	0.198	0.194	0.218	0.219
	MAE_σ	0.039	**0.041**	0.036	**0.034**	0.044	0.044	0.029	0.046
DS_{30}	$\overline{MAE}$	0.142	**0.123**	**0.140**	0.169	0.185	0.216	0.225	0.198
	MAE_σ	0.047	**0.029**	**0.056**	0.057	0.048	0.048	0.028	0.030
DSS_5	$\overline{MAE}$	0.212	**0.152**	0.163	**0.145**	0.163	0.241	0.295	0.382
	MAE_σ	0.067	**0.045**	0.058	**0.036**	0.019	0.115	0.088	0.057
DSS_{15}	$\overline{MAE}$	0.135	**0.123**	0.154	0.140	**0.134**	0.201	0.231	0.286
	MAE_σ	0.020	**0.013**	0.046	0.020	**0.021**	0.069	0.059	0.094
DSS_{30}	$\overline{MAE}$	**0.132**	0.134	0.135	0.139	**0.133**	**0.132**	0.208	0.320
	MAE_σ	**0.026**	0.012	0.015	0.011	**0.009**	**0.021**	0.065	0.111

Tabla 4.7: Media de MAE y desviación en los conjuntos de prueba de cada una de las técnicas para datasets Sector y Sector Simplificado

Dataset	GA	GACE $45-5$	GACE $40-10$	GACE $35-15$	GACE $25-25$	GACE $15-35$	GACE $10-40$	CE
DP_5	0.023	**0.022**	**0.020**	**0.020**	0.023	0.045	0.028	0.039
DP_{15}	0.017	**0.011**	**0.011**	**0.013**	0.015	0.023	0.023	0.067
DP_{30}	0.044	**0.017**	**0.016**	**0.017**	0.019	0.018	0.026	0.042
DPS_5	0.020	**0.019**	**0.018**	0.025	**0.018**	0.023	0.027	0.303
DPS_{15}	0.017	0.016	**0.011**	**0.012**	0.015	0.060	0.109	0.434
DPS_{30}	0.031	0.027	**0.021**	**0.021**	0.026	0.028	0.040	0.298

Tabla 4.8: Media del error sMAPE en los datasets DP y DPS por cada una de las técnicas.

Dataset	GA	GACE $45-5$	GACE $40-10$	GACE $35-15$	GACE $25-25$	GACE $15-35$	GACE $10-40$	CE
DS_5	0.199	0.199	0.202	0.204	**0.198**	**0.197**	0.340	0.484
DS_{15}	0.333	**0.240**	0.251	**0.217**	**0.240**	0.365	0.463	0.545
DS_{30}	**0.202**	**0.186**	0.244	0.210	0.205	0.318	0.378	0.445
DSS_5	0.237	**0.201**	**0.234**	0.296	0.327	0.375	0.396	0.355
DSS_{15}	**0.301**	**0.256**	0.322	0.344	0.311	0.374	0.375	0.361
DSS_{30}	0.306	**0.221**	0.328	**0.299**	0.346	0.343	0.387	0.387

Tabla 4.9: Media del error sMAPE en los datasets DS y DSS por cada una de las técnicas.

lo hace para 7 de los 12 casos. Tanto $GACE_{10-40}$ como CE no obtienen nunca uno de los dos mejores resultados en ningún caso.

Con estos últimos resultados, y en dicha ampliación, se realizó una pequeña comparativa con las mejores configuraciones de GACE encontradas y diversas técnicas de la literatura. Estas técnicas fueron:

- ◇ C4.5 [Quinlan 14]: es un algoritmo que genera árboles de decisión a partir de un conjunto de ejemplos proporcionados. Dicho árbol de decisión se construye desde arriba hacia abajo. Es una mejora del algoritmo ID3.
- ◇ Linear Decreasing Weight - Particle Swarm Optimization (LDWPSO) [Sousa 04]: Este método es un algoritmo PSO que usa como función de velocidad la función de pesos decrecientes lineales, de donde recibe su nombre.
- ◇ Steady-State Genetic Algorithm for Extracting Fuzzy Classification Rules (SGERD) [Mansoori 08]: Algoritmo genético de tipo steady-state (no crea una nueva población al término de la generación). Las generaciones durante las que se aplica el algoritmo son finitas y están ligadas a la dimensión del problema. La selección de los individuos no es aleatoria y sólo sobreviven los mejores. La función fitness utilizada por este método se basa en el criterio de evaluación de reglas. SGERD se detiene cuando ningún hijo es incluido en la población.

Los resultados de esta comparativa se muestran en la Tabla 4.10. En los casos donde se pretende predecir la congestión en un punto (datasets DP y DPS), C4.5

Dataset	GACE $45-5$	GACE $40-10$	GACE $35-15$	C4.5	LDWPSO	SGERD
DP_5	0.022	**0.020**	**0.020**	**0.002**	0.100	0.043
DP_{15}	**0.011**	**0.011**	0.013	**0.004**	0.100	0.043
DP_{30}	0.017	**0.016**	0.017	**0.004**	0.089	0.043
DPS_5	0.019	**0.018**	0.025	**0.018**	**0.013**	0.604
DPS_{15}	0.016	**0.011**	**0.012**	0.033	0.027	0.108
DPS_{30}	**0.027**	**0.021**	**0.021**	0.052	**0.027**	0.029
DS_5	**0.199**	0.202	0.204	**0.048**	0.606	0.202
DS_{15}	0.240	0.251	**0.217**	**0.140**	0.604	0.396
DS_{30}	**0.186**	0.244	0.210	**0.176**	0.602	0.395
DSS_5	**0.201**	**0.234**	0.296	0.362	0.362	0.684
DSS_{15}	**0.256**	**0.322**	0.344	0.328	0.423	0.766
DSS_{30}	**0.221**	0.328	0.299	0.310	0.400	**0.270**

Tabla 4.10: Comparativa de las configuraciones de GACE con los algoritmos C4.5, LDWPSO, y SGERD

es mejor que GACE en 3 de los 6 casos de estudio de estos conjuntos de datos, mientras que GACE está presente como una de las dos mejores técnicas en todos los datasets. En el caso de los datasets de tipo sector (DS y DSS), GACE obtiene mejores resultados que los métodos con los que se compara en 3 de los 6 casos de estudio propuestos. Por otro lado, LDWPSO y SGERD obtienen los mejores valores en dos y un único caso respectivamente.

Por otro lado, es importante conocer qué porcentaje de cada tipo de congestión ha predicho con éxito cada una de las diferentes configuraciones. Para ello, las Figuras 4.9, 4.10, 4.11, y 4.12 muestran el porcentaje de instancias clasificadas correctamente por cada tipo de congestión que se ha predicho en este trabajo. Cada barra indica los valores de media y desviación de cada ejecución. Por lo tanto, hay 8 barras por cada imagen. Cada barra se denota por su GA_{tam} en el eje X. Por lo tanto, de izquierda a derecha los valores van desde el GA puro hasta el CE.

Si se analizan estas figuras, podemos decir que:

◇ En la mayoría de los casos, las configuraciones del GACE mejoran o igualan

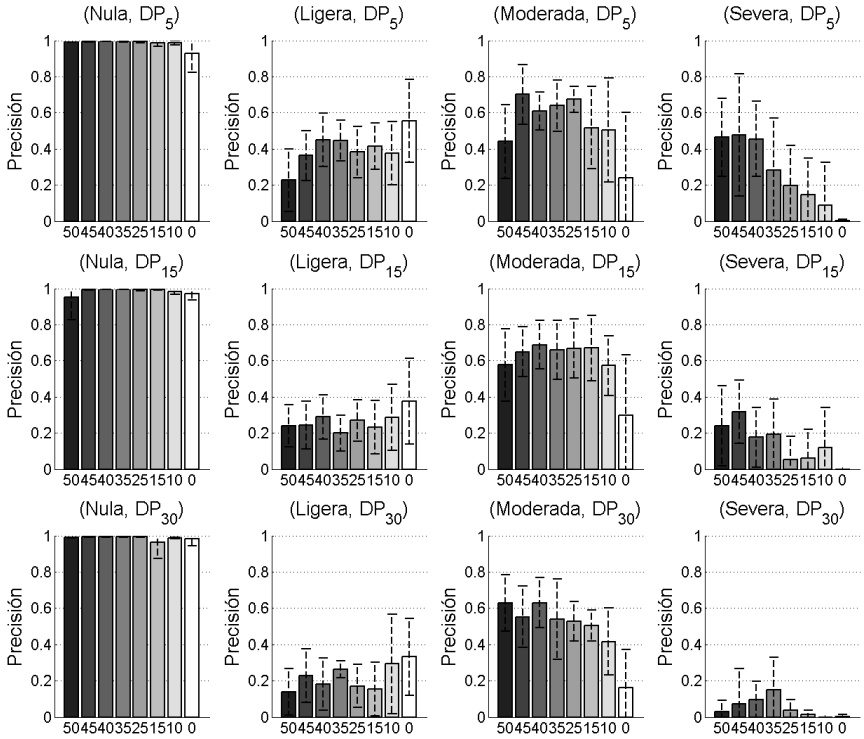

Figura 4.9: Porcentaje de instancias correctamente clasificadas por cada clase en los datasets DP_5 (arriba), DP_{15} (centro), y DP_{30} (abajo).

el rendimiento obtenido por las técnicas que lo forman, GA y CE.

◇ De las dos partes, el GA puro es el que muestra unos valores de error más cercanos a los obtenidos por GACE en la mayoría de los conjuntos.

◇ Los mejores valores de predicción se obtienen cuando el margen de tiempo es el más pequeño de los presentados (5 minutos) y cuando se quiere predecir la congestión en un punto (datasets DP y DPS).

◇ Para los datasets DS y DSS ocurre todo lo contrario: una previsión de 30 minutos obtiene buenos resultados.

◇ Para los conjuntos de datos simplificados, se obtiene prácticamente el mismo valor para cualquier tipo de congestión, siendo los mejores resultados los obtenidos en DPS_5.

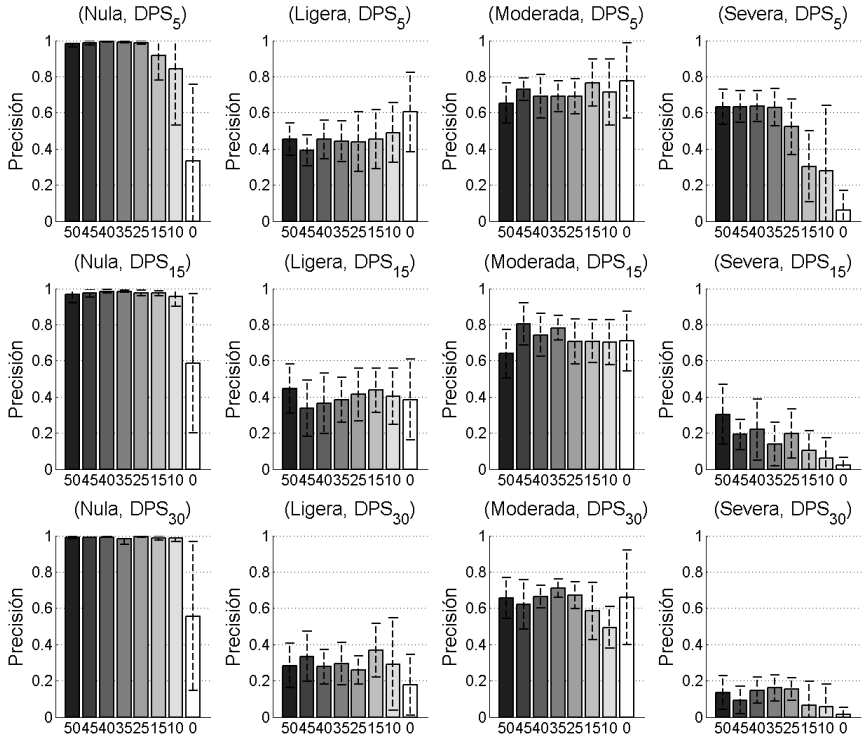

Figura 4.10: Porcentaje de instancias correctamente clasificadas por cada clase en los datasets DPS_5 (arriba), DPS_{15} (centro), y DPS_{30} (abajo).

- ◇ Si se predice congestión en un segmento (datasets DS y DSS), se obtienen buenos resultados a la hora de predecir congestión Severa en cualquier dataset.

- ◇ Predecir congestión Severa en un segmento obtiene valores significativamente mejores que predecir congestión Nula en conjuntos de datos simplificados, especialmente en DSS_5 y DSS_{15}, donde los resultados entre los diferentes tipos de congestión son más estables.

Con los datos de la Tabla 4.6, donde se utiliza MAE como métrica del error, se ha utilizado el conocido test t de Student para mostrar las diferencias estadísticas que existen entre cada una de las configuraciones y algoritmos utilizados. En dicho test, utilizamos los siguientes valores:

1. $(++)$ en el caso de que los resultados del algoritmo A_1 sean significativamente mejores a los del algoritmo A_2.

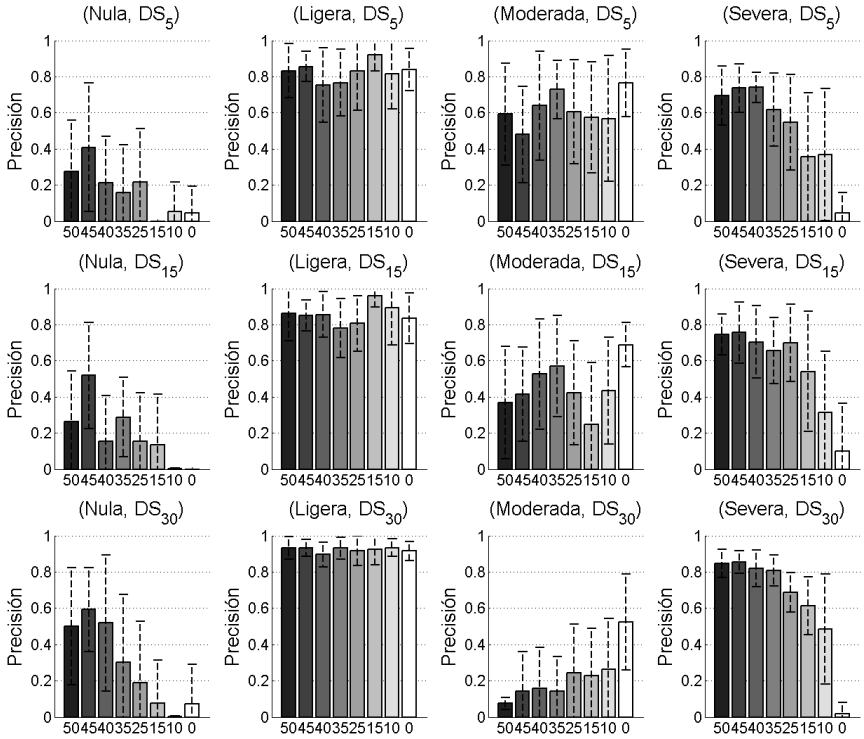

Figura 4.11: Porcentaje de instancias correctamente clasificadas por cada clase en los datasets DS_5 (arriba), DS_{15} (centro), y DS_{30} (abajo).

2. $(+)$ en el caso de que los resultados del algoritmo A_1 sean mejores a los del algoritmo A_2.

3. $(-)$ en el caso de que los resultados del algoritmo A_1 sean peores estadísticamente a los obtenidos por el algoritmo A_2.

4. $(--)$ en el caso de que los resultados del algoritmo A_1 sean significativamente peores estadísticamente a los obtenidos por el algoritmo A_2.

El cálculo de este valor se realiza siguiendo la Ecuación 4.8:

$$t = \frac{\overline{X_1} - \overline{X_2}}{\sqrt{\frac{(n_1-1)SD_1^2+(n_2-1)SD_2^2}{n_1+n_2-2}\frac{n_1+n_2}{n_1 n_2}}} \tag{4.8}$$

Donde $\overline{X_i}$, SD_i y n_i, son la media, la desviación y el número de ejecuciones de cada técnica. En nuestro caso, n_i está definido con un valor 10 para todas las

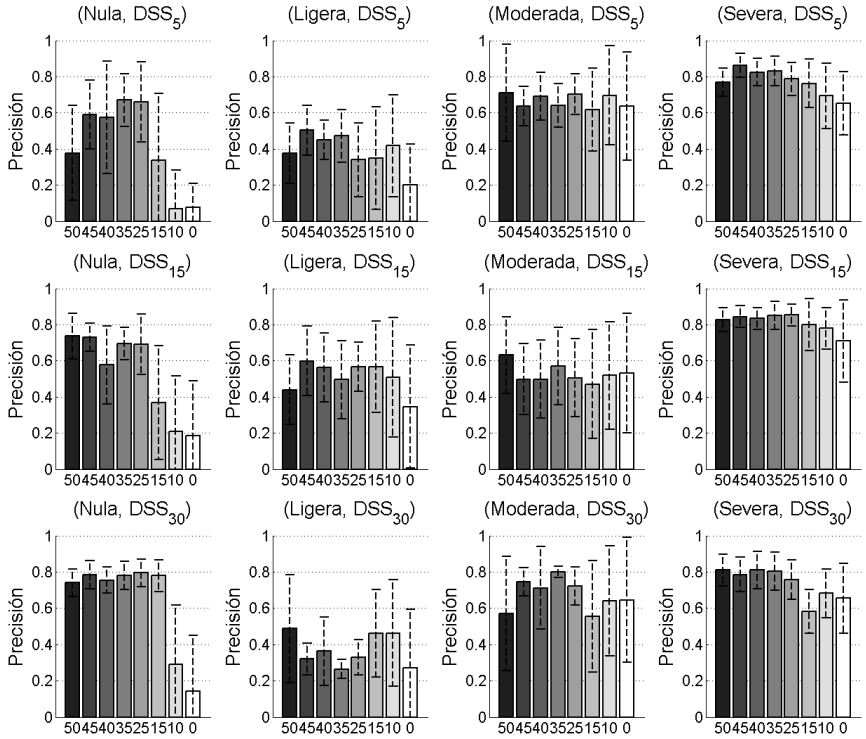

Figura 4.12: Porcentaje de instancias correctamente clasificadas por cada clase en los datasets DSS_5 (arriba), DSS_{15} (centro), y DSS_{30} (abajo).

técnicas, mientras que los valores medios y de desviación son los obtenidos en la Tabla 4.6. El intervalo de confianza escogido es de 0.95.

Como resultado, se incluye en las Tablas 4.11 y 4.12 el porcentaje de veces que los resultados un algoritmo son mejores $(+)$ o significativamente mejores que los de otro $(++)$, quedando demostrado lo observado en la experimentación anterior. Realizando un análisis de dichas tablas, podemos decir que:

- ◇ La configuración $GACE_{45-5}$ es mejor o significativamente mejor que el GA puro en más del 80 % de los casos.
- ◇ Por otro lado, $GACE_{45-5}$ es significativamente mejor $(++)$ que el CE puro en todos los casos expuestos.
- ◇ Comparando configuraciones de la propuesta, utilizando $GACE_{40-10}$ se mejora el rendimiento obtenido por $GACE_{45-5}$ en un 25 % de los casos totales,

mientras que la aplicación de $GACE_{45-5}$ supera a $GACE_{40-10}$ en el 50 % de los 48 casos.

- ◇ Por otro lado, la configuración $GACE_{35-15}$ supera a $GACE_{40-10}$ en el mismo porcentaje total de casos en los que $GACE_{40-10}$ es mejor a $GACE_{45-5}$, un 25 %.

- ◇ A su vez, $GACE_{45-5}$ mejora a $GACE_{35-15}$ en un porcentaje superior que a $GACE_{40-10}$ con casi un 60 % de los casos de estudio.

- ◇ Comparando los resultados obtenidos por las técnicas puras, GA es mejor o significativamente mejor que CE en el 100 % de los casos.

- ◇ La aplicación de un GA puro es únicamente significativamente mejor $(++)$ que las configuraciones del algoritmo $GACE_{15-35}$,$GACE_{10-40}$, y CE, y obtiene mejor resultado $(+)$ en un porcentaje menor del 50 % cuando es comparada con el resto de configuraciones, siendo el porcentaje más alto obtenido al compararlo con la configuración $GACE_{35-15}$ con aproximadamente un 42 %.

- ◇ $GACE_{10-40}$ y $GACE_{15-35}$ obtienen un resultado de $++$ comparándolos con CE en un 58 % de los casos y, a su vez, $GACE_{15-35}$ obtiene un resultado de $+$ y $++$ comparándolo con la configuración $GACE_{10-40}$ en más del 80 % de los casos totales.

- ◇ CE sólo es mejor que las configuración $GACE_{15-35}$ en un 25 % de los casos y que $GACE_{10-40}$ en un 16 %.

- ◇ Por último, CE no obtiene ningún valor $++$ cuando es comparado con alguno de las configuraciones propuestas.

Para ilustrar lo obtenido por el test de Student, las Figuras 4.13, 4.14, 4.15 y 4.16 muestran los resultados generales. Los colores verde muestran cuando un algoritmo es mejor que otro, siendo el color verde más oscuro el correspondiente al valor $(+)$ y el más claro a $(++)$. Igualmente, cuando un algoritmo es peor estadísticamente que otro, se muestran colores rojo que corresponden a $(-)$ el más oscuro

	GA	$GACE_{45-5}$	$GACE_{40-10}$	$GACE_{35-15}$	$GACE_{25-25}$	$GACE_{15-35}$	$GACE_{10-40}$	CE
GA	X	8,33	33,33	41,67	33,3	58,33	16,67	16,67
$GACE_{45-5}$	50	X	41,67	25	33,3	33,33	16,67	0
$GACE_{40-10}$	41,67	8,33	X	50	25	58,33	25	16,67
$GACE_{35-15}$	33,3	25	25	X	41,67	41,67	33,33	16,67
$GACE_{25-25}$	41,67	25	25	33,3	X	33,3	33,33	25
$GACE_{15-35}$	16,67	8,33	8,33	16,67	16,67	X	66,67	16,67
$GACE_{10-40}$	8,33	0	0	0	0	8,33	X	25
CE	0	0	0	0	0	25	16,67	X

Tabla 4.11: Porcentaje de veces en los que los resultados de un algoritmo son mejores que los obtenidos por otro $(+)$

	GA	$GACE_{45-5}$	$GACE_{40-10}$	$GACE_{35-15}$	$GACE_{25-25}$	$GACE_{15-35}$	$GACE_{10-40}$	CE
GA	X	0	0	0	0	16,67	66,67	83,33
$GACE_{45-5}$	33,33	X	8,33	33,33	33,33	50	83,33	100
$GACE_{40-10}$	16,67	16,67	X	0	33,33	33,33	75	83,33
$GACE_{35-15}$	25	8,33	0	X	25	41,67	66,67	83,33
$GACE_{25-25}$	16,67	0	8,33	0	X	41,67	66,67	75
$GACE_{15-35}$	0	0	0	0	8,33	X	16,67	58,33
$GACE_{10-40}$	0	0	0	0	0	0	X	58,33
CE	0	0	0	0	0	0	0	X

Tabla 4.12: Porcentaje de veces en los que los resultados de un algoritmo son significitivamente mejores que los obtenidos por otro $(++)$

y el más claro a $(--)$. Por último, cuando los resultados de ambos algoritmos son iguales, se muestran casillas de color negro .

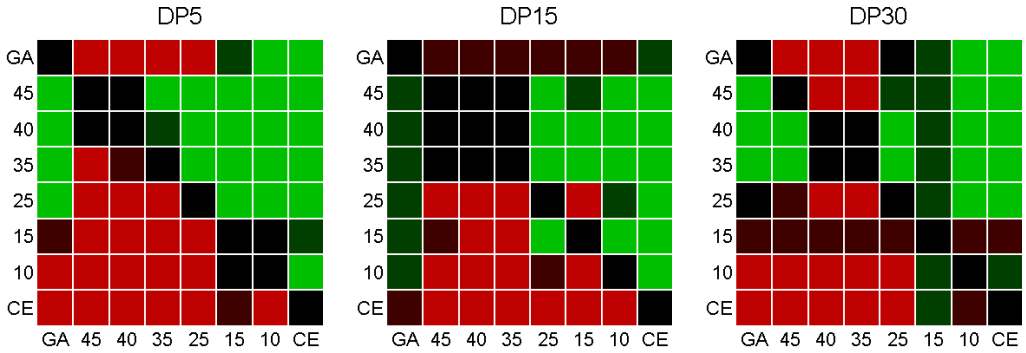

Figura 4.13: Resultados del Test de Student para los datasets DP.

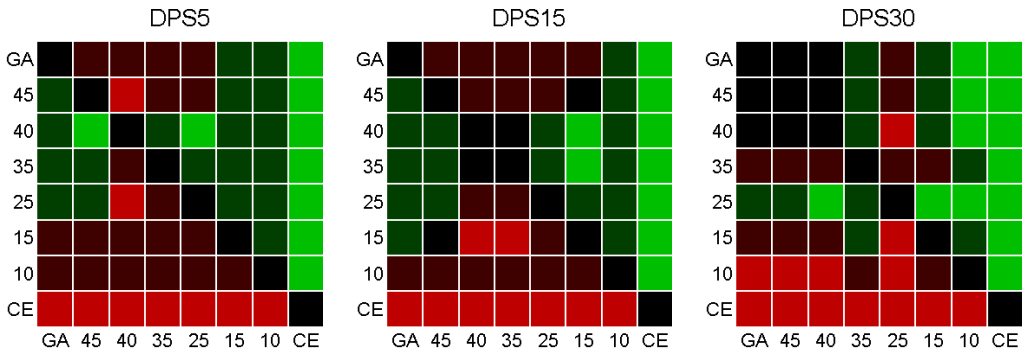

Figura 4.14: Resultados del Test de Student para los datasets DPS.

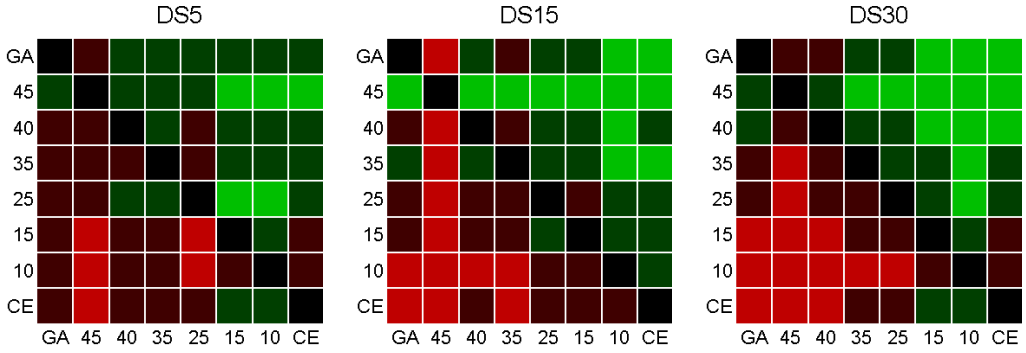

Figura 4.15: Resultados del Test de Student para los datasets DS.

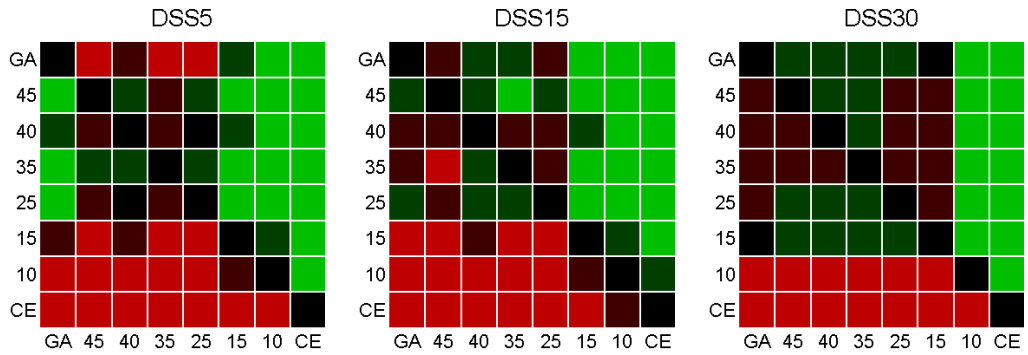

Figura 4.16: Resultados del Test de Student para los datasets DSS.

	GA	GACE $45-5$	GACE $40-10$	GACE $35-15$	GACE $25-25$	GACE $15-35$	GACE $10-40$	CE
Nula	4.16	**2.66**	**2.91**	3.08	3.08	5.75	6.75	7.58
Ligera	4.50	4.25	4.83	5.33	5.33	**3.66**	**3.41**	4.66
Moderada	5.33	4.66	**3.58**	**3.16**	3.91	5.75	5.41	4.16
Severa	2.91	**2.33**	**2.58**	3.16	4.16	6.25	6.75	7.83
Total	4.22	**3.47**	**3.47**	3.68	4.12	5.35	5.58	6.06

Tabla 4.13: Media de posición para cada técnica cuando predice diferentes niveles de congestión. Los dos mejores valores están resaltados para cada caso.

En estas figuras queda patente lo obtenido en la Tabla 4.6, donde las técnicas $GACE_{45-5}$ y $GACE_{40-10}$ obtienen por lo general los mejores resultados. Se puede comprobar como en todas estas figuras, su parte baja suele ser roja, lo que nos indica que las técnicas con bajo nivel de GA son normalmente mejoradas por el resto.

Por otro lado, la Tabla 4.13 muestra la posición que ocupa cada técnica en todos los datasets, es decir, la posición media que ocupa cuando son ordenadas por mejor resultado obtenido en cada conjunto. Los resultados muestran que:

- $GACE_{45-5}$ es el mejor situado en 2 de los 4 tipos de congestión, cuando esta toma los valores Nula y Severa.
- $GACE_{40-10}$ se encuentra entre los dos mejores casos en los tipos de congestión Nula, Moderada y Severa, ”fallando”únicamente en la congestión Ligera.

	Nula	Ligera	Moderada	Severa
DP_5	$GACE_{40-10}$	CE	$GACE_{45-5}$	$GACE_{45-5}$
DP_{15}	$GACE_{35-15}$	CE	$GACE_{40-10}$	$GACE_{45-5}$
DP_{30}	$GACE_{25-25}$	CE	GA	$GACE_{35-15}$
DPS_5	$GACE_{40-10}$	CE	CE	$GACE_{40-10}$
DPS_{15}	$GACE_{40-10}$	GA	$GACE_{45-5}$	GA
DPS_{30}	$GACE_{25-25}$	$GACE_{15-35}$	$GACE_{35-15}$	$GACE_{35-15}$
DS_5	$GACE_{45-5}$	$GACE_{15-35}$	CE	$GACE_{40-10}$
DS_{15}	$GACE_{45-5}$	$GACE_{15-35}$	CE	$GACE_{45-5}$
DS_{30}	$GACE_{45-5}$	$GACE_{10-40}$	CE	$GACE_{45-5}$
DSS_5	$GACE_{35-15}$	$GACE_{45-5}$	GA	$GACE_{45-5}$
DSS_{15}	GA	$GACE_{45-5}$	GA	$GACE_{25-25}$
DSS_{30}	$GACE_{25-25}$	GA	$GACE_{35-15}$	GA

Tabla 4.14: Mejores técnicas que predicen congestión para cada conjunto de datos.

- ◇ $GACE_{35-15}$ acompaña a $GACE_{40-10}$ en un único caso, siendo mejor que este para predecir congestión Moderada.

- ◇ Las configuraciones $GACE_{15-35}$ y $GACE_{10-40}$ se alzan como mejor técnica a utilizar en el caso de predecir congestión Ligera.

- ◇ CE queda como última técnica a elegir en 2 de los 4 casos, con congestión Nula y Severa, mientras que en los dos restantes la última posición es ocupada por las configuraciones $GACE_{35-15}$ y $GACE_{25-25}$ en el caso de congestión Ligera, y $GACE_{15-35}$ en el caso de congestión Moderada.

- ◇ En líneas generales, $GACE_{45-5}$ y $GACE_{40-10}$ están mejor situados para ser utilizados que el resto de configuraciones, mientras que GA sería la cuarta opción por encima de las configuraciones $GACE_{15-35}$, $GACE_{10-40}$ y CE.

Para obtener más detalles de la experimentación realizada en este apartado, se presenta la Tabla 4.14, donde se muestran las mejores técnicas para cada dataset y congestión. Analizando dicha tabla, podemos llegar a las siguientes conclusiones:

- ◇ GACE obtiene mejores resultados que el resto de técnicas en 32 de 48 casos, lo que representa que es superado por uno de los dos métodos puros, (GA ó CE) en sólo el 33 % de los casos.

- ◇ Haciendo una división por tipo de congestión, GACE es la técnica a utilizar en congestión Nula en 11 de 12 casos y en Severa en 10 de 12, mientras que, para congestión Ligera y Moderada, GACE es la técnica elegida en 6 y 5 casos respectivamente.

- ◇ Destaca que $GACE_{15-35}$ sea el elegido en 3 casos de congestión Ligera en diferentes datasets dado los resultados obtenidos a lo largo de la experimentación, donde las técnicas con mayor población en su parte genética obtenían mejores resultados.

- ◇ CE, pese a su mal rendimiento durante toda la experimentación, es el algoritmo elegido en un total de 8 de 42 casos, centrándose en los datasets completos de Punto y Sector en los tipos de congestión Ligera y Moderada. GA, por su parte, obtiene los mismos resultados: es elegida 8 veces, sin un patrón establecido.

- ◇ La técnica que obtiene mejores resultados un mayor número de veces ha sido GACE con población $GACE_{45-5}$, con un total de 12 de 48 casos, de los cuáles 3 son en congestión Nula, 2 en Ligera, 2 en Moderada y 5 en Severa, siendo en este último tipo de congestión la más elegida.

- ◇ En el caso opuesto, $GACE_{10-40}$ es elegida únicamente una vez, en detección de congestión Ligera en un dataset de tipo Sector, lo que la convierte en la técnica menos elegida de todo el conjunto presentado.

- ◇ Finalmente, y como dato puntual, todas las técnicas han sido elegidas al menos una vez a lo largo de los 48 casos de estudio.

Como conclusiones a este apartado de la experimentación realizada, podemos determinar los siguientes hechos:

- ◇ El uso del algoritmo propuesto con población genética en el intervalo $GA_{tam} = [35, 45]$ aporta mejores resultados que el resto de configuraciones utilizadas y los algoritmos puros, ya sea utilizando error MAE o $sMAPE$.

- ◇ En la comparativa con las técnica propuestas, GACE, utilizando las configuraciones mencionadas en el punto anterior, obtiene uno de los dos errores con menor valor en todos los datasets.

- ◇ En lo que a instancias de los diferentes tipos de congestión bien clasificadas se refiere, GACE mejora o iguala el rendimiento obtenido por los algoritmos puros GA y CE.

- ◇ Estadísticamente hablando, las configuraciones $GACE_{45-5}$, $GACE_{40-10}$ y $GACE_{35-15}$ se encuentran entre las técnicas más destacadas obteniendo resultados mejores o significativamente mejores frente al resto de configuraciones y técnicas.

- ◇ Mientras que $GACE_{45-5}$ es la mejor técnica para poder predecir congestión Nula y Severa, $GACE_{15-35}$ y $GACE_{35-15}$ lo son para congestiones de tipo Ligera y Moderada respectivamente.

- ◇ Por último, y en términos generales, de los 48 casos de estudio, GACE, cualquiera que sea su configuración, obtiene resultados por encima de los métodos puros en 32 de los casos, siendo por tanto superado únicamente en un 33 % de los casos totales.

4.2 Aplicación del algoritmo a funciones de optimización

Esta sección se encarga de recoger toda la experimentación relacionada con la aplicación del algoritmo propuesto en esta tesis a funciones de optimización. Las funciones que se han utilizado quedan definidas en la Sección 4.2.1. Para poder contar con unos valores por defecto y realizar las modificaciones de las que se ha hablado en la Sección 3.6, se realiza un estudio de los parámetros en la Sección 4.2.2. Por

último, la comparativa con otras técnicas de la literatura se encuentra contenida en la Sección 4.2.3.

4.2.1 Funciones utilizadas

Un total de 24 funciones de optimización extraídas de la plataforma Comparing Continuous Optimisers (COCO) se han utilizado para esta parte de la experimentación. Esta plataforma se utiliza en el workshop Black Box Optimization Benchmarking (BBOB). En dicho workshop, que tiene lugar durante las conferencias internacionales GECCO y CEC desde 2009 en adelante, se aplican algoritmos propuestos por investigadores a las funciones anteriores con diferente número de dimensiones. El objetivo de esta competición es probar estos algoritmos en funciones benchmark, de cara a encontrar el mejor valor posible para la función en un tiempo predefinido.

Estas funciones están libres de ruido y son de diferentes tipos. Para no entorpecer la lectura del documento, así como mantener la atención en el capítulo de experimentación que nos ocupa, el nombre, tipo y fórmula de estas funciones se adjuntan en el Apéndice 6. Las funciones se denotarán a lo largo de este capítulo con la nomenclatura F_i, donde i toma valores desde 1 hasta 24, siendo estas de los siguientes tipos:

- ◇ De la función F_1 hasta la función F_5, las funciones son de tipo separable.
- ◇ Se cuenta con un total de 4 funciones con condicionamiento bajo o moderado, correspondientes a los índices F_6 hasta F_9.
- ◇ Las funciones desde la F_{10} hasta la función F_{14} será de tipo condicionamiento elevado y unimodales.
- ◇ Las funciones multimodales con estructura global adecuada se encuentran en las funciones desde F_{15} hasta F_{19}.
- ◇ Por último, las funciones de F_{20} hasta F_{24} serán de tipo multimodal con estructura global débil.

Por tanto, contamos en total 5 funciones Separables, 4 funciones con condicionamiento bajo o moderado, 5 con condicionamiento elevado y unimodales, 5

funciones multimodales con estructura global adecuada y, por último, 5 funciones multimodales con estructura global débil.

La razón de usar estas funciones, como ya se ha expuesto, es demostrar la adaptabilidad de GACE a problemas de optimización genéricos, para así demostrar que puede utilizarse de una manera sencilla en otros marcos de trabajo, garantizando unos niveles de calidad adecuados en comparación con técnicas de optimización encontradas en el estado del arte. Por otra parte, se busca reducir el número de parámetros a introducir por el usuario, como es el tamaño de la población, o la tasa de actualización del CE, y estableciendo unos valores por defecto al resto de parámetros de la configuración, los cuales deben garantizar un grado de calidad en las soluciones.

4.2.2 Estudio de los parámetros del algoritmo

En esta sección se detalla el estudio de los diferentes parámetros de los que consta el algoritmo de forma que se pueda reducir su número. En primer lugar, se buscará definir unos valores por defecto para dichos parámetros. Para poder fijar el tamaño de la población de cara a futuras experimentaciones e investigación, este modifica su valor dependiendo del valor dado a la dimensión del problema. Por tanto, se ha definido el tamaño de población como $POB_{tam} = c \cdot dim$, donde c es una constante que puede tomar los valores $c = \{2, 5, 10, 20\}$, y dim la dimensión, que corresponde con el tamaño del individuo.

Para cada uno de los posibles valores de POB_{tam}, se tienen en cuenta diferentes porcentajes asociados al tamaño de población genética GA_{tam}. Este parámetro, definido como p_{ga}, toman el 25 %, 50 % y 75 % del tamaño total de la población POB_{tam}, es decir, $p_{ga} = \{0.25, 0.5, 0.75\}$. En la Tabla 4.15 se resumen los valores por defecto que se han tomado para los diferentes parámetros utilizados en esta experimentación.

En primer lugar, es necesario determinar el valor de la constante c para cada posible valor de la dimensión dim, dado que esto determinará el tamaño de la población en cada una de las dimensiones en las que vamos a trabajar. Siendo dim una variable que puede tomar los valores $dim = \{5, 10, 20, 40\}$, el tamaño de la población quedará definido en el intervalo $POB_{tam} \in [10, 800]$. A su vez, tener

Parámetros	Valor
POB_{tam}	$\{2, 5, 10, 20\} \cdot dim$
Porcentaje de individuos del GA	$p_{ga} = \{0.25, 0.5, 0.75\}$
Porcentaje de individuos del CE	$p_{ce} = 1 - p_{ga}$
Parámetros del GA	$p_c = 0.9$, $p_m = \frac{1}{dim}$, $\alpha = 0.5$
Parámetros de CE	$L_r = 0.7$, $p_{up} = 1$

Tabla 4.15: Valores de los diferentes parámetros utilizados en la experimentación

una primera aproximación al valor que tiene que tomar p_{ga} nos ofrecerá una idea general de los tamaños de las sub-poblaciones. Teniendo en cuenta los valores que p_{ga} puede tomar, podemos llegar a la conclusión de que el tamaño de la población genética se encuentra en el intervalo $GA_{tam} \in [3, 600]$ y la población a la que se aplica la CE estaría contenida en el intervalo $CE_{tam} \in [2, 600]$. Un ejemplo para entender lo expuesto: Para $dim = 5$ y $c = 2$, GA_{tam} tomará los valores $GA_{tam} = \{3, 5, 8\}$ y $CE_{tam} = \{7, 5, 2\}$, siendo CE_{tam} calculada como se describió en la experimentación sobre congestión, es decir $CE_{tam} = POB_{tam} - GA_{tam}$.

Se realiza en primera instancia, por tanto, un estudio de estos dos parámetros. Para que los resultados que se han obtenido se puedan entender de una manera sencilla, la Tabla 4.16 provee los rankings prmedio obtenidos para cada dimensión. Se ha utilizado como condición de parada para esta experimentación un total de 25000 evaluaciones. Las funciones utilizadas han sido optimizadas entre $R_{min} = -5$ y $R_{max} = 5$ tal y como se indica en la documentación provista por la competición BBOB. Lo que buscamos con este estudio es encontrar qué tamaño de población y qué porcentaje p_{ga} aproximado es necesario para obtener los mejores resultados posibles para la mayoría de las funciones. Los valores resaltados representan los dos mejores rankings promedio en cada una de las dimensiones consideradas.

Para $dim = 5$, los mejores valores se consiguen en configuraciones con $c = \{5, 10\}$ y $p_{ga} = \{0.5, 0.75\}$. En el caso de $dim = 10$ los tres mejores valores se dan con la constante $c = 5$. Para el resto de dimensiones ($dim = \{20, 40\}$), $c = 2$ con todos los valores posibles de p_{ga} consiguen todos mejores rankings. Basándonos en estos resultados, podemos decir que, para dimensiones menores o iguales a $dim = 10$, la constante debería tomar el valor $c = 5$, mientras que, para

	$c=2$			$c=5$			$c=10$			$c=20$		
p_{ga}	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
$Rank_{dim=5}$	8.38	8.33	6.33	5.25	**4.50**	5.83	6.13	5.50	**5.08**	8.29	6.25	5.13
$Rank_{dim=10}$	5.17	5.58	5.63	**4.50**	4.75	**4.54**	7.38	6.29	6.29	11.00	8.79	8.04
$Rank_{dim=20}$	**2.63**	**3.29**	3.92	5.17	4.42	5.46	8.63	6.58	6.88	11.67	9.88	9.50
$Rank_{dim=40}$	3.25	**2.67**	**2.75**	5.25	4.33	5.38	9.00	7.33	7.54	11.25	9.83	9.42
$Rank_{general}$	5.00	4.75	5.00	3.75	**3.00**	5.00	8.50	6.50	6.25	11.50	10.25	8.25

Tabla 4.16: Ranking promedio de los diferentes algoritmos utilizados por dimensión y valor de p_{ga}

el resto de casos, la constante debería tomar el valor $c = 2$. De una manera más formal, la fórmula para determinar el tamaño de la población se definiría como en la Ecuación 4.9.

$$POB_{tam} = \begin{cases} 5 \cdot dim \text{ si } dim \leq 10 \\ 2 \cdot dim \text{ si } dim > 10 \end{cases} \tag{4.9}$$

Dados estos valores, y habiendo tanteado el valor óptimo para el porcentaje de GA_{tam}, se realiza un estudio más exhaustivo del parámetro p_{ga}. Además, no hay que olvidar que el algoritmo divide la población en dos sub-poblaciones, donde en una se ejecuta un GA mientras que en otra se aplica un CE. Como se explicó en secciones anteriores, CE cuenta con un parámetro, n_{up}, que determina el número de individuos que utiliza para actualizar la media y la desviación previa con el fin de crear nuevos individuos. El estudio de este parámetro, junto con el porcentaje p_{ga}, resulta fundamental para poder encontrar la configuración óptima. Por tanto, al igual que p_{ga} de encarga del tamaño de la población genética GA_{tam}, se hace necesaria la creación de un porcentaje p_{up} que ayudará a determinar el valor del número de individuos n_{up}. El valor de n_{up} dependerá a su vez del tamaño de la población de CE (CE_{tam}), dado que el número de individuos utilizados para actualizar los valores de media y desviación no podrá superar el tamaño total de la población en la que se aplicará la CE.

Para este estudio, p_{up} tomará los valores $p_{up} = \{0.2, 0.4, 0.6, 0.8, 1\}$, mientras que $p_{ga} = \{0, 0.05, 0.1 \cdots, 1\}$. Se estudia cada posible par (p_{ga}, p_{up}) para determinar la mejor combinación posible. Un ejemplo: $(p_{ga}, p_{up}) = (0.4, 0.2)$ denotará una ejecución en el que el 40 % de los individuos de la población POB_{tam} formarán

la sub-población genética, mientras que el restante 60 % formará la sub-población dedicada a CE. De este último 60 %, el 20 % de los mejores individuos serán los utilizados para actualizar los valores de media y desviación asociados al método de entropía cruzada.

Para mostrar el resultado, se ha hecho uso de mapas de calor que denotan el rango que se obtiene para cada combinación de parámetros en cada dimensión. Cuanto más oscuro sea el cuadrado, mejor será el ranking. La Figura 4.17 muestra dichos mapas de calor mientras que la Tabla 4.17 resume los valores tomados para los parámetros en esta parte de la experimentación.

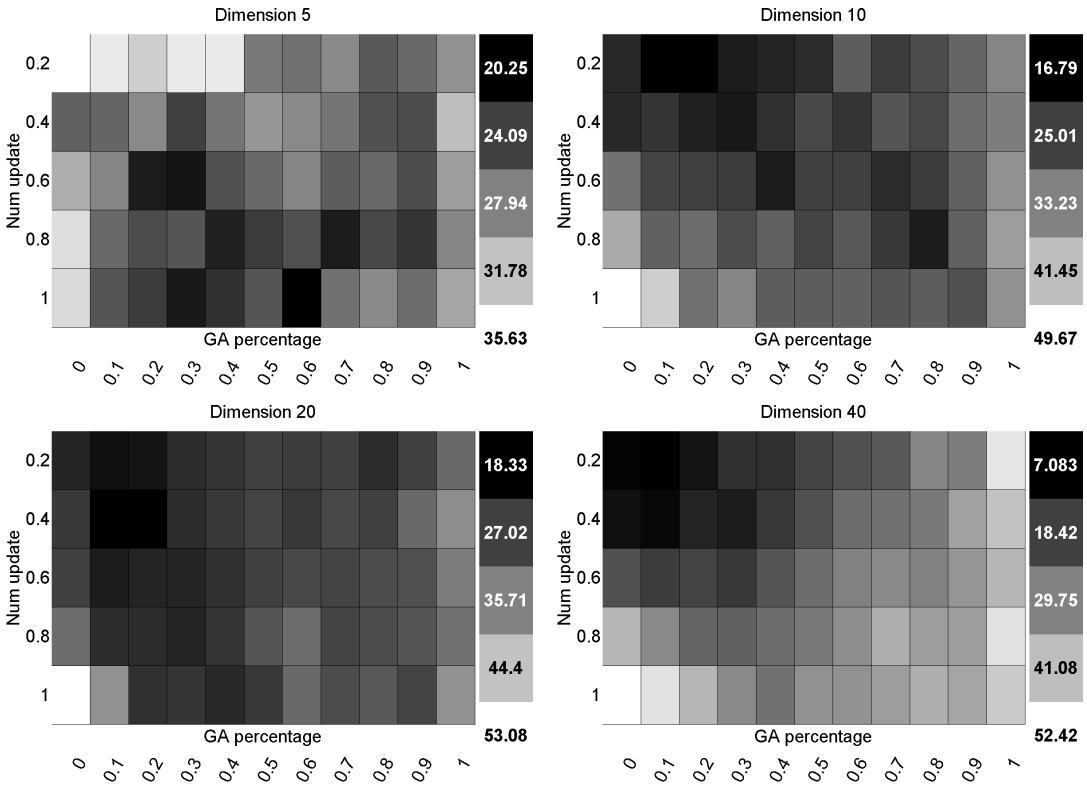

Figura 4.17: Ranking promedio de cada dupla de valores utilizada en los posibles valores de dimensión dim. El eje x contiene los porcentajes posibles de p_{ga} mientras que el eje y se exponen los valores del porcentaje p_{up}. En la parte superior se muestran los gráficos de calor de $dim = 5$ (izquierda) y $dim = 10$ (derecha). En la parte inferior, los correspondientes a $dim = 20$ (izquierda) y $dim = 40$ (derecha).

Con respecto a las figuras, el eje X muestra los porcentajes de GA_{tam} mientras

Parámetros	Valor
POB_{tam}	Ecuación 4.9
Porcentaje de inviduos del GA	$p_{ga} = \{0, 0.05, 0.1 \ldots 1\}$
Nº de inviduos utilizados para actualizar	$p_{up} = \{0.2, 0.4 \ldots 1\}$
Condición de parada	T_{max} = 25000

Tabla 4.17: Valores de los diferentes parámetros utilizados en la segunda parte de la experimentación

que el eje Y muestra los valores de p_{up} utilizados. Las figuras superiores muestran los resultados para las dimensiones 5 y 10 mientras que las figuras inferiores lo hacen para las dimensiones 20 y 40. En cada figura se muestra una leyenda (colocada a la derecha) con la escala de grises y los valores a los que corresponde cada color.

Comenzando por la dimensión 5 ($dim = 5$), los mejores valores se encuentran en los intervalos $p_{up} \in [0.6, 1]$ y $p_{ga} \in [0.3, 0.7]$, mientras que los peores valores están situados en la fila de $p_{up} = 0.2$ y en la columna de $p_{ga} = 0$. Para el resto de dimensiones, los mejores resultados quedan concentrados en los intervalos $p_{up} \in [0.2, 0.6]$ y $p_{ga} \in [0, 0.3]$. Cuando el valor de la dimensión aumenta, es fácil ver que las mejores combinaciones de estos parámetros se obtienen con valores bajos de ambos, dado que es donde se encuentran los cuadros más oscuros. Además, cuanto mayor es la dimensión, más aumenta la diferencia entre los mejores y los peores rankings. Una de las combinaciones obtenidas que obtienen un buen rendimiento en dimensiones altas y razonablemente bueno en bajas es $p_{up} = 0.4$ y $p_{ga} = 0.1$. Se utilizarán estos valores para la comparativa entre técnicas existentes en la literatura en la siguiente sección.

4.2.3 Comparativa de los resultados

Esta sección contiene la comparativa entre el método propuesto con los parámetros elegidos durante los diferentes estudios realizados anteriormente y las técnicas existentes en la literatura. Estos parámetros y el valor que toman quedan recogidos en la Tabla 4.18.

Con la idea de evitar que tanto el algoritmo propuesto como los de la literatura usados en la comparativa siempre converjan al mismo valor, se han utilizado un

Parámetros	Valor
POB_{tam}	Ecuación 4.9
Porcentaje de individuos del GA	$p_{ga} = 0.1$
Porcentaje de individuos del CE	$p_{ce} = 0.9$
Parámetros del GA	$p_c = 0.9$, $p_m = \frac{1}{dim}$, $\alpha = 0.5$
Parámetros del CE	$L_r = 0.7$, $p_{up} = 0.4$
Condición de parada	$T_{max} = 25000$

Tabla 4.18: Parámetros utilizados para la comparativa entre el algoritmo y los métodos del estado del arte tras los estudios realizados

total de 15 instancias, todas ellas con valores óptimos diferentes. Las técnicas usadas para la comparativa pertenecen al workshop BBOB 2013 realizado durante la conferencia GECCO del mismo año. Las técnicas incluyen un GA con codificación real, dos tipos de GA celulares, un GA generacional y un un método Hill-Climber. Se han escogido los siguientes métodos para tener una comparativa de calidad, dado que se utilizaron también dichas funciones. Otro motivo es el de poder alcanzar un mejor rendimiento que técnicas como un GA como es el de codificación real, una técnica con el mismo enfoque que la propuesta, utilizando varias poblaciones (en el caso del GA celular) o una técnica realmente simple (como el Hill-Climber). Las configuraciones usadas para estos métodos son las indicadas por sus autores en sus artículos originales. A continuación se mencionan las características de las técnicas:

- ◇ GA con codificación real basado en proyección aumentada (PRCGA). Este algoritmo cuenta con cinco operadores: selección por torneo, cruce blend-α, mutación no uniforme, proyección y un mecanismo para evitar el estancamiento. La probabilidad de cruce se establece a 0.8, la de mutación a 0.15 y el tamaño del torneo a 3. Para el proceso de experimentación, las conclusiones o más detalles, consultar [Sawyerr 13].

- ◇ La implementación de la Evolución Diferencial (DE) aplicada en [Pošík 12] en su forma estándar, con 'best/1'como operador de mutación.

- ◇ Los GA celulares (CGA) son una variedad de los GA paralelos. Se definen como GAs en los que la población es dividida en dos sub-poblaciones semi-separadas. En este caso, cada individuo es su propia sub-población. Dos de estos algoritmos se utilizan para la comparativa. Uno de ellos, llamado Grid, se implementa en su forma estándar tal y como se describe en [Alba 09], mientras que el otro es un "ring" CGA, desarrollado en [Holtschulte 13].

- ◇ Se utiliza un GA generacional (GGA) con selección por ranking para la comparativa.

- ◇ Por último, un algoritmo Hill-Climber (Hill) [Jacobson 04] es una técnica de optimización iterativa que pertenece a la familia de las técnicas de búsqueda local. En esta comparativa se utiliza la técnica desarrollada en [Holtschulte 13].

Las Tablas 4.19–4.22 muestran el error medio. Se ha considerado el error como el mejor fitness obtenido menos el óptimo de la instancia utilizada. Es decir $error = F_{mejor} - F_{opt}$. Como condición de parada para las técnicas utilizadas, se utiliza el mismo número de evaluaciones que se han empleado anteriormente, 25000. Los valores en negrita representan los dos mejores valores obtenidos para cada función. Los resultados con un asterisco $*$ indican el mejor valor para el error alcanzado para cada función.

Para $dim = 5$, GACE obtiene uno de los dos resultados destacados en 11 de las 24 funciones, y el más cercano al óptimo de esos resultados en 3 de ellas. El mayor número de resultados destacados lo obtiene la técnica DE con 16 de 24, siendo la mejor en 14 de ellos. $PRCGA$, $Ring$ y GGA muestran un rendimiento similar con 5, 5 y 7 valores entre los mejores resultados respectivamente. Las técnicas con peor rendimiento han sido $Grid$, que sólo obtiene el un resultado superior en dos ocasiones, y $Hill$, que obtiene el mejor resultado en 4 funciones.

En el caso de $dim = 10$, $Ring$ y $Grid$ obtienen 1 y 4 de los resultados destacados, siendo las técnicas que menos tienen en esta dimensión. Centrándonos en DE, obtiene uno de los dos resultados destacados en 13 de 24 funciones, siendo la técnica que obtiene el valor más cercano al óptimo en 8 de esos casos. Por otro lado, $PRCGA$ mejora su rendimiento hasta contar con 6 valores destacados entre

	GACE	DE	GGA	Grid	Hill	Ring	PRCGA
F_1	**0.00e+00***	**7.08e-09**	1.19e-04	1.03e-04	1.58e-05	2.28e-04	2.70e-05
F_2	**6.62e-06**	**6.13e-09***	1.70e+00	3.35e+00	2.09e+00	1.35e+00	1.81e-03
F_3	**2.72e-03**	2.00e-01	5.44e-02	7.20e-02	**1.49e-02***	1.24e-01	4.15e-01
F_4	1.93e+00	1.89e+00	**9.50e-02**	2.15e-01	**3.99e-02***	2.46e-01	8.67e-01
F_5	**-1.02e-14***	**-1.02e-14***	**-1.02e-14***	**-1.02e-14***	**-1.02e-14***	**-1.02e-14***	**2.24e+00**
F_6	2.48e-01	**7.58e-09***	1.53e-01	4.44e-01	**9.16e-02**	2.80e-01	1.60e+00
F_7	6.12e-01	**4.49e-09***	**1.38e-01**	6.98e-01	9.74e-01	2.04e-01	2.26e-01
F_8	3.54e+00	**5.29e-01***	1.47e+00	1.69e+00	1.40e+00	**8.25e-01**	1.50e+00
F_9	3.85e+00	**5.30e-01***	2.39e+00	3.78e+00	3.47e+00	1.76e+00	**1.73e+00**
F_{10}	5.04e+02	**2.61e-02***	2.88e+02	9.87e+02	8.43e+02	**2.09e+02**	2.06e+03
F_{11}	2.37e+01	**1.38e-03***	**6.42e+00**	3.54e+01	1.38e+01	8.07e+00	1.99e+01
F_{12}	7.05e+00	**1.71e+00***	7.48e+01	6.38e+01	5.73e+01	3.82e+02	**5.78e+00**
F_{13}	**3.76e+00**	**1.34e-03***	1.34e+01	1.15e+01	3.12e+01	8.34e+00	4.59e+00
F_{14}	**5.05e-04**	**9.94e-08***	3.14e-03	1.00e-02	9.80e-03	5.81e-03	2.90e-03
F_{15}	**3.11e+00**	**2.03e+00***	3.61e+00	1.12e+01	1.41e+01	5.78e+00	6.96e+00
F_{16}	**3.14e-01***	9.87e-01	**4.66e-01**	1.01e+00	1.59e+00	6.51e-01	5.98e-01
F_{17}	**6.24e-03**	**1.38e-04***	6.15e-02	4.27e-01	1.01e+00	1.40e-01	8.71e-02
F_{18}	**1.18e-01**	**2.24e-02***	3.31e-01	2.44e+00	2.43e+00	6.10e-01	2.72e-01
F_{19}	**1.77e-01**	4.01e-01	3.83e-01	6.96e-01	6.36e-01	6.14e-01	**3.04e-02***
F_{20}	5.63e-01	2.48e-01	**1.94e-01**	4.80e-01	5.04e-01	**1.68e-01***	8.66e-01
F_{21}	2.35e+00	8.38e-01	6.50e-01	1.99e+00	2.80e+00	**6.06e-04***	**4.73e-01**
F_{22}	1.62e+00	1.09e+00	**1.04e+00**	2.78e+00	1.59e+00	**1.49e-01***	2.32e+00
F_{23}	9.93e-01	1.19e+00	1.00e+00	**8.84e-01***	9.60e-01	**9.06e-01**	9.33e-01
F_{24}	9.81e+00	**9.76e+00**	1.11e+01	1.21e+01	1.08e+01	1.21e+01	**7.71e+00***

Tabla 4.19: Media del error de las técnicas para $dim = 5$

los resultados. GACE también mejora, obteniendo 14 de los mejores resultados y siendo en 6 de ellos la que menos error ha obtenido. Esto la coloca como la segunda técnica en lo que a resultados se refiere para esta dimensión.

Para la experimentación con una dimensión $dim = 20$, las técnicas que obtienen más resultados destacados son $PRCGA$ y $GACE$, con 9 y 18 entre los mejores valores respectivamente. De entre las 18 funciones en las que $GACE$ destaca, ob-

	GACE	DE	GGA	Grid	Hill	Ring	PRCGA
F_1	**0.00e+00***	**8.36e-09**	1.82e-03	1.42e-03	1.68e-04	1.05e-02	4.21e-06
F_2	**1.53e-05**	**8.27e-09***	1.36e+01	1.65e+01	1.30e+01	4.27e+01	1.07e-01
F_3	3.02e+00	1.12e+00	**6.38e-01**	1.09e+00	**8.14e-02***	2.54e+00	1.54e+00
F_4	7.88e+00	3.08e+00	**1.27e+00**	1.74e+00	**2.65e-01***	4.08e+00	2.47e+00
F_5	**5.95e-13**	**5.86e-15***	**5.86e-15***	**5.86e-15***	**5.86e-15***	**5.86e-15***	9.52e+00
F_6	3.10e+00	**1.06e-02***	**1.36e+00**	5.17e+00	2.49e+00	4.48e+00	1.76e+00
F_7	**2.18e+00**	**1.82e-01***	2.33e+00	7.08e+00	6.11e+00	3.23e+00	4.08e+00
F_8	9.58e+00	**2.38e+00**	1.48e+01	2.23e+01	**2.13e+00***	1.02e+01	1.12e+01
F_9	2.52e+01	**5.53e+00**	1.15e+01	7.33e+01	5.55e+01	1.27e+01	**4.76e+00***
F_{10}	**3.55e+03***	1.31e+04	**4.47e+03**	1.81e+04	1.84e+04	6.61e+03	6.40e+03
F_{11}	4.64e+01	8.39e+01	**3.15e+01***	9.79e+01	6.87e+01	4.26e+01	**4.00e+01**
F_{12}	**3.13e+00***	1.91e+02	1.28e+03	2.17e+03	1.33e+03	1.17e+04	**1.71e+01**
F_{13}	**6.17e+00**	**2.38e+00***	1.73e+01	3.65e+01	1.80e+01	3.93e+01	1.15e+01
F_{14}	**3.60e-03**	**7.49e-04***	1.61e-02	2.70e-02	1.44e-02	3.24e-02	4.54e-03
F_{15}	**6.79e+00***	3.85e+01	2.27e+01	5.93e+01	7.61e+01	3.87e+01	**1.28e+01**
F_{16}	**1.44e+00***	1.01e+01	**2.40e+00**	6.39e+00	6.28e+00	3.41e+00	2.43e+00
F_{17}	**1.50e-02**	**2.98e-03***	2.89e-01	3.11e+00	3.25e+00	9.11e-01	2.75e-01
F_{18}	**1.82e-01***	**4.22e-01**	1.32e+00	1.06e+01	1.61e+01	3.11e+00	1.09e+00
F_{19}	**1.27e+00**	2.81e+00	2.37e+00	3.46e+00	3.21e+00	3.00e+00	**3.30e-02***
F_{20}	1.29e+00	**3.55e-01***	**5.16e-01**	7.17e-01	6.65e-01	7.13e-01	1.30e+00
F_{21}	3.74e+00	2.77e+00	**1.92e+00**	6.03e+00	1.48e+01	**4.36e-01***	4.01e+00
F_{22}	7.80e+00	**2.21e+00**	4.22e+00	1.47e+01	6.59e+00	**7.51e-01***	4.79e+00
F_{23}	1.66e+00	1.90e+00	1.41e+00	1.47e+00	**1.32e+00***	**1.38e+00**	1.45e+00
F_{24}	**3.56e+01**	5.03e+01	4.67e+01	6.39e+01	5.59e+01	5.56e+01	**2.93e+01***

Tabla 4.20: Error medio de las técnicas para $dim = 10$

tiene el resultado más cercano al óptimo en 8 de ellas. Como se puede observar, hay una diferencia entre los resultados obtenidos en las anteriores dimensiones y la que se trata actualmente, ya que se ve aumentado el rendimiento de $GACE$ a la hora de obtener mejores resultados. Para este valor de dimensión, DE obtiene uno de los dos mejores resultados sólo en 7 de 24 funciones, siendo el mejor en 5 de ellas. En el caso de $Grid$ y $Ring$, siguen obteniendo los resultados más alejados

	GACE	DE	GGA	Grid	Hill	Ring	PRCGA
F_1	**5.13e-05**	**8.62e-06***	2.65e-02	2.12e-02	1.33e-03	2.56e-01	1.82e-04
F_2	**1.13e-02***	**3.94e-02**	1.10e+02	2.23e+02	6.54e+01	8.76e+02	3.16e+01
F_3	9.87e+00	9.45e+01	**5.71e+00**	7.61e+00	**8.55e-01***	2.33e+01	6.20e+00
F_4	2.63e+01	9.07e+01	**9.90e+00**	1.09e+01	**1.53e+00***	3.34e+01	1.20e+01
F_5	2.85e-09	**-3.61e-15***	**6.46e-14**	**6.46e-14**	**6.46e-14**	**6.46e-14**	1.19e+01
F_6	**1.47e+01***	1.80e+02	**1.49e+01**	4.60e+01	2.66e+01	6.66e+01	2.57e+01
F_7	**1.32e+01**	2.13e+01	**1.22e+01***	4.14e+01	3.38e+01	2.79e+01	1.72e+01
F_8	**2.64e+01**	**2.59e+01***	7.72e+01	9.49e+01	4.80e+01	8.26e+01	6.46e+01
F_9	**3.66e+01**	**2.81e+01***	5.29e+01	9.44e+01	7.62e+01	1.01e+02	4.03e+01
F_{10}	**3.54e+04**	3.27e+05	4.26e+04	8.23e+04	4.28e+04	4.68e+04	**3.11e+04***
F_{11}	**8.86e+01***	2.46e+02	9.76e+01	2.04e+02	2.21e+02	1.51e+02	**9.04e+01**
F_{12}	**3.13e+02**	3.89e+02	2.46e+04	1.90e+04	1.42e+03	2.60e+05	**1.10e+02***
F_{13}	**2.16e+01**	**1.27e+01***	6.67e+01	6.10e+01	2.46e+01	1.49e+02	2.19e+01
F_{14}	**1.31e-02***	2.33e-02	5.25e-02	6.40e-02	2.65e-02	3.05e-01	**1.88e-02**
F_{15}	**2.11e+01***	1.47e+02	9.16e+01	2.29e+02	2.61e+02	1.48e+02	**4.26e+01**
F_{16}	**2.53e+00***	2.88e+01	7.73e+00	1.36e+01	1.35e+01	1.09e+01	**5.45e+00**
F_{17}	**1.75e-01***	**5.70e-01**	6.69e-01	8.13e+00	1.12e+01	3.29e+00	7.40e-01
F_{18}	**8.47e-01***	5.39e+00	4.76e+00	2.91e+01	4.24e+01	1.20e+01	**2.70e+00**
F_{19}	**2.90e+00**	6.21e+00	5.24e+00	7.78e+00	7.35e+00	6.74e+00	**2.06e-01***
F_{20}	2.29e+00	2.66e+00	**9.07e-01**	1.17e+00	**8.82e-01***	1.44e+00	1.63e+00
F_{21}	**4.61e+00**	5.73e+00	5.01e+00	1.20e+01	1.86e+01	**1.54e+00***	1.12e+01
F_{22}	9.15e+00	8.80e+00	**7.55e+00**	2.38e+01	1.31e+01	**3.33e+00***	7.93e+00
F_{23}	2.43e+00	3.45e+00	2.66e+00	**2.42e+00**	**2.10e+00***	2.51e+00	2.42e+00
F_{24}	**1.45e+02**	1.60e+02	1.51e+02	2.38e+02	2.51e+02	2.07e+02	**1.11e+02***

Tabla 4.21: Error medio de las técnicas para $dim = 20$

del óptimo también en esta dimensión.

Por último, para la dimensión $dim = 40$, las técnicas con mayor diferencia entre el óptimo y el valor obtenido son GGA y $Ring$, con 3 de 24 resultados, y DE, con sólo un valor entre los mejores. $Grid$ sigue estando entre las técnicas que obtienen mayores diferencias con 4 resultados mientras $Hill$ mejora con respecto a la dimensión anterior, con 9 de 24 posibles resultados. En este caso, la propuesta

	GACE	DE	GGA	Grid	Hill	Ring	PRCGA
F_1	**3.98e-08***	7.33e+00	4.25e-01	2.62e-01	1.17e-02	3.93e+00	**7.27e-03**
F_2	**8.52e+00***	3.12e+04	2.26e+03	2.71e+03	6.74e+02	2.35e+04	**8.04e+01**
F_3	4.31e+01	3.98e+02	4.21e+01	**3.99e+01**	**7.13e+00***	1.43e+02	4.51e+01
F_4	6.65e+01	6.83e+02	**5.77e+01**	6.27e+01	**1.12e+01***	2.05e+02	7.83e+01
F_5	**2.46e-01**	**1.34e-14***	**1.34e-14***	**1.34e-14***	**1.34e-14***	**1.34e-14***	3.99e+01
F_6	**6.77e+01***	3.44e+03	1.35e+02	4.42e+02	1.56e+02	3.95e+02	**1.06e+02**
F_7	**3.97e+01***	4.92e+02	9.57e+01	1.70e+02	1.57e+02	1.56e+02	**6.61e+01**
F_8	**5.44e+01***	3.39e+03	2.59e+02	2.45e+02	**1.06e+02**	8.31e+02	1.82e+02
F_9	**4.28e+01***	2.88e+03	2.48e+02	4.33e+02	**9.38e+01**	1.04e+03	1.89e+02
F_{10}	**1.12e+05***	2.94e+06	1.76e+05	3.73e+05	1.54e+05	4.18e+05	**1.24e+05**
F_{11}	**1.90e+02***	7.06e+02	2.91e+02	4.86e+02	4.56e+02	3.51e+02	**1.99e+02**
F_{12}	**5.44e+02***	2.44e+07	4.22e+05	3.36e+05	3.29e+04	5.40e+06	**8.63e+03**
F_{13}	**7.06e+01**	9.69e+02	2.25e+02	1.90e+02	**5.79e+01***	5.89e+02	9.96e+01
F_{14}	**2.65e-02***	1.91e+01	3.40e-01	2.54e-01	**5.57e-02**	4.02e+00	8.27e-02
F_{15}	**8.99e+01***	6.04e+02	2.96e+02	7.58e+02	8.44e+02	5.81e+02	**1.47e+02**
F_{16}	**1.05e+01**	4.48e+01	1.52e+01	2.67e+01	2.40e+01	2.29e+01	**9.34e+00***
F_{17}	**1.99e-01**	7.75e+00	1.79e+00	1.34e+01	2.11e+01	7.84e+00	**1.49e+00***
F_{18}	**1.01e+00***	3.01e+01	8.69e+00	5.14e+01	7.92e+01	2.73e+01	**5.98e+00**
F_{19}	**5.90e+00**	9.50e+00	8.02e+00	1.29e+01	1.16e+01	9.68e+00	**2.50e-01***
F_{20}	3.14e+00	8.55e+01	1.57e+00	**1.52e+00**	**8.77e-01***	2.89e+00	1.81e+00
F_{21}	5.98e+00	4.82e+01	6.35e+00	7.15e+00	1.69e+01	**5.27e+00**	**4.95e+00***
F_{22}	**7.83e+00**	3.91e+01	9.00e+00	2.30e+01	1.74e+01	**5.43e+00***	9.49e+00
F_{23}	4.06e+00	5.97e+00	4.24e+00	**3.33e+00**	**2.99e+00***	3.66e+00	4.39e+00
F_{24}	**3.87e+02**	6.36e+02	**3.80e+02***	7.68e+02	8.19e+02	6.18e+02	4.42e+02

Tabla 4.22: Error medio de las técnias para $dim = 40$

es la mejor técnica con 19 de 24 valores destacados, 12 de los cuales son el valor más cercano al óptimo, seguido de $PRCGA$ con 13.

Para resumir todos estos resultados, en un total de 96 casos de estudio (24 funciones por 4 dimensiones utilizadas), $GACE$ obtiene uno de los dos mejores valores en 62 casos, de los cuales alcanza el menor error en 29 de ellos. La segunda mejor técnica ha sido DE, con un resultado entre los mejores en 37 de los 96

	GACE	DE	GGA	Grid	Hill	Ring	PRCGA
$Rank_{dim=5}$	3.69	**2.44**	3.60	5.5	4.81	3.83	4.12
$Rank_{dim=10}$	3.19	**3.02**	3.23	5.85	4.60	4.69	3.42
$Rank_{dim=20}$	**2.46**	4.21	3.5	5.35	4.46	5.12	2.89
$Rank_{dim=40}$	**2.08**	6.33	3.54	4.71	3.75	4.71	2.87
$Rank_{all}$	**2.85**	4	3.47	5.35	4.41	4.59	3.33

Tabla 4.23: Resultados del test de Friedman para cada dimensión

casos, siendo el mejor en 28 de ellos. Para asegurarnos de que las diferencias que se han encontrado en lo que a rendimiento se refiere son o no significativas, se hace necesario el uso de test estadísticos. Para ello, se han utilizado dos test estadísticos siguiendo las indicaciones propuestas en [Derrac 11].

En primer lugar, se aplica el test de Friedman [Derrac 11] para conocer si hay diferencias significativas entre los métodos comparados. La Tabla 4.23 muestra los rankings proporcionados por este test no paramétrico para cada uno de los algoritmos en cada dimensión y globalmente sobre todas las dimensiones.

Como se puede ver en dicha tabla, el algoritmo propuesto obtiene el mejor ranking en $dim = 20$ y $dim = 40$, y en términos globales. Se puede apreciar como la diferencia entre el ranking obtenido por nuestra técnica y los diferentes algoritmos con los que se ha comparado es mayor cuando tanto las dimensiones como, por tanto, la complejidad del problema aumentan.

Los test post-hoc de Holm [Holm 79] y Finner [Finner 93] se aplican a los resultados obtenidos por el test de Friedman usando DE como método de control en $dim = \{5, 10\}$ y $GACE$ en el resto de dimensiones. La Tabla 4.24 muestra los p-values ajustados devueltos por dichos tests para cada dimensión y técnica. Los valores que se ofrecen están redondeados a tres decimales. Para resaltar las diferencias significativas, los p-values por debajo de 0.1 se muestran en negrita.

De manera general, $GACE$ es significativamente mejor que el resto de métodos excepto uno, $PRCGA$. En este caso, aunque la diferencia no es signifiticativa, el nivel de confianza es alto, concretamente un 88 %.

Si nos fijamos en los resultados obtenidos en cada dimensión, para $dim = 5$, DE mejora significativamente a $Grid$, $Hill$ y $Ring$. En $dim = 10$, DE mejora sig-

	$dim = 5$		$dim = 10$			$dim = 20$		$dim = 40$		Global	
	Holm	Finner	Holm	Finner		Holm	Finner	Holm	Finner	Holm	Finner
GACE	0.09	0.054	1.577	0.799	DE	**0.015**	**0.007**	**0**	**0**	**0.001**	**0**
GGA	0.09	0.061	1.577	0.799	GGA	0.189	0.113	**0.039**	**0.023**	**0.097**	**0.058**
Grid	**0**	**0**	**0**	**0**	Grid	**0**	**0**	**0**	**0**	**0**	**0**
Hill	**0.001**	**0.001**	0.062	**0.022**	Hill	**0.005**	**0.003**	**0.022**	**0.011**	**0**	**0**
Ring	0.076	**0.037**	**0.038**	**0.022**	Ring	**0**	**0**	**0**	**0**	**0**	**0**
PRCGA	**0.027**	**0.013**	1.577	0.673	PRCGA	0.483	0.483	0.204	0.204	0.128	0.128

Tabla 4.24: Estadísticas de los test de Holm y Finner para cada dimensión y técnica

nificativamente a $Grid$, $Hill$ y $Ring$ según Finner, mientras que, según Holm, sólo mejora a $Grid$ y $Ring$. Finalmente, para las dimensiones restantes ($dim = 20, 40$), ambos tests indican que $GACE$ mejora significativamente a todos los métodos menos a $PRCGA$. Concrétamente, en la dimensión $dim = 40$, se puede encontrar el valor más cercano entre los métodos $PRCGA$ y $GACE$.

Con los resultados aportados por los experimentos realizados a lo largo de esta sección, podemos llegar a las siguientes conclusiones:

- ◇ Comenzando por el tamaño de la población general (POB_{tam}), se ha definido este con valores, en general, bajos, dado que va desde 25 individuos en el caso de $dim = 5$ hasta 80 en el caso de $dim = 40$, siendo en este último donde se dan los resultados más óptimos.

- ◇ Con respecto a los porcentajes que determinan la población genética utilizada (p_{ga}) y la actualización de la media y desviación necesarias para CE (p_{up}), un valor pequeño del primero, entre el 0 y el 30 %, y uno intermedio en el caso del segundo, entre 20 % y 60 % de la población de CE, ofrecen un rendimiento óptimo tanto en dimensiones bajas como en altas.

- ◇ Independientemente del tipo de función, $GACE$ trabaja de manera muy eficiente en dimensiones altas, dejando atrás a técnicas del estado del arte en optimización tan conocidas como DE.

- ◇ Sin embargo, en dimensiones bajas ($dim = \{5, 10\}$), aunque su rendimiento se asemeja al de DE, es superado por esta técnica.

- ◇ En general, y para todas las dimensiones utilizadas, $GACE$ sería la técnica elegida dada su posición en el ranking promedio de resultados.
- ◇ Estadísticamente, supera en dimensiones altas a aquellas técnicas con las que se le ha comparado.

Que el que llore el último llore peor.
Y que los últimos no sean los de siempre.

Marta Martinez Carro

5

Conclusions and Future Work

This chapter contains the conclusions from the results obtained in the experimentation studies carry out along this dissertation. First, Section 5.1 contains the reflections we can extract from the results obtained by applying the proposal to congestion forecasting. After that, in Section 5.2, the conclusions about the different parameter studies made by using GACE in benchmark functions are exposed. Finally, the answer to the hypothesis made at the beginning of this memory, and future work lines are contained in Section 5.3.

5.1 Conclusions about Congestion Forecasting

As it has been said along the dissertation, one of the aims to create GACE was using it to optimize the different parts of the systems that form a hierarchy of FRBSs, i.e. variables, labels and rules. This optimization was carrying out with the objective of predict congestion at a point or segment in a road. To do that, a total of 12 datasets are created with data of a freeway in California. The prediction horizon takes values of 5, 15 and 30 minutes.

The experimentation is made focusing in the different sizes of the sub-populations in which GA and CE are applied respectively. Different types of datasets are used to prove the performance of the developed algorithm and the hierarchical systems. Two error metrics are used in order to check this performance. With the first one, MAE, GACE with higher size of GA population obtains better results than those with the same sub-population size or more CE_{size} than GA_{size}. Also, these results improve the obtained ones by GA and CE. While $GACE_{40-10}$ obtains one of the two best values in 7 out of 12 cases, $GACE_{45-5}$ does the same in 8 out of 12 cases. The use of other performance metric is necessary due to MAE does not take into account the differences between the type of predicted congestion and the expected one. As it has been said in previous sections, sMAPE is used to avoid this problem. With the results obtained with this metric, it can be said that the conclusions obtained with MAE metrics are correct: GACE with bigger GA_{size} than CE_{size} obtains the best errors. $GACE_{45-5}$ obtains in this case 9 out of 12 best values while $GACE_{40-10}$ and $GACE_{35-15}$ obtain 7 out of 12 best values each one.

Using sMAPE results, GACE is compared with three literature methods. Looking at the results, it can be said that the proposed technique is a competitive technique, but it can improve its performance in several datasets.

Without taking into account the kind of dataset, the experiment about what configuration of the proposed technique is better for each kind of congestion was made. The conclusions are that $GACE_{45-5}$ is the best option in Free and Severe congestion, while $GACE_{40-10}$, despite not to be the first best option, it achieves the second place in 3 out of 4 types of congestion. In case of configurations with low value of GA_{tam}, $GACE_{15-35}$ and $GACE_{10-40}$ are the best techniques to forecast Light congestion. In case of the pure configurations, GA has obtained the fourth position as algorithm to use, while CE has been the last option.

Now, taking into account the type of dataset and predicted congestion value, GACE is the best technique to use in 32 out of 48 cases of study. As it was mentioned before, GACE configurations are the best to predict Null (11 out of 12) and Severe (10 out of 12) congestion, while it is the best technique for forecast Light and Moderate congestions in 6 and 5 out of 12 cases. Despite its bad performance, CE is the chosen algorithm in 8 out of 42, as well as GA.

Finally, about the kind of dataset and prediction horizon, Point Datasets (DP and DPS) and a short-time horizon (5 minutes) obtain the best error values. In the case of Sector Datasets (DS and DSS) is the opposite: the best values are obtained in the 30-minutes horizon.

5.2 Conclusions about Function Optimization

The other part of the experimentation has the aim to prove the performance of the proposed algorithm in different types of optimization functions. With this objective in mind, a total of 24 noise-free functions have been taken from a platform used in different international conferences.

The most important issue in this part of the experimentation was to choose the value from different important parameters as the size of the population, the sizes of the sub-populations, learning rate for the CE part, number of individuals needed to update the means and deviations vectors, and so on. This is important, not only for this problem, but the future works in other different themes. Although it is important the tunning of the parameters depending of the problem, with this, default values for the parameters that assure a good performance in general are found and studied.

First, the size of the population is calculated based on a constant and the dimension of the problem, given as results $c = 5$ if dimension is less or equal to 10, and $c = 2$ if dimension is greater than 10. After that, heat graphics are used to delimitate the values of the size of GA sub-population, and, in the other hand, the percentage of individuals to update the different parts of CE. At the end, intervals $p_{ga} \in [0.3, 0.7]$ for GA size, and $p_{up} = [0.6, 1]$ obtain good performance for low dimensions, while $p_{ga} \in [0, 0.3]$ for GA size, and $p_{up} = [0.2, 0.6]$ do the same for high dimensions. With this, we can assure a greater number of individuals in GA_{pob}, and practically the whole CE_{pob} used for update if the dimension of the problem is high, and the opposite, i.e. small values for both parameters, if the dimension of the problem is high. For the comparative with literature methods, $p_{up} = 0.4$ and $p_{ga} = 0.1$ were chosen due to their good performance in high dimensions.

As a summary of the results obtained, the proposal improves its results while the dimension of the functions is growing. For $dim = 5$, GACE obtains one of the

two best values in 11 out of 24 functions, while with $dim = 40$, it obtains one of the two best values in 19 out of 24 while the rest of the techniques are getting worse results, in special the Differential Evolution. For the total 96 cases of study, GACE obtains one of the two best values in 62 cases, while it is the best of all in 29 of them. The second technique who obtains better results is the Differential Evolution, with 37 out of 96, and the best so far in 28 out of 37.

Statistic tests as Friedman, Holm and Finner are used to assure the results obtained. Using DE as control algorithm for dimension 5 and 10, and GACE for the rest of the dimensions, the result of the tests prove that GACE is significantly different than the rest of the algorithms except for PRCGA, which obtains the closer value to it in $dim = 40$.

5.3 Improvements and future lines of work

Along this chapter, the conclusions extracted from the different experimentations made have been remarked. The algorithm and the FRBSs have obtained a good performance in forecasting congestion, regardless of the type of dataset used, but especially in Point and Section datasets. After that, with the parameters selected in experimentation studies, the algorithm can reach a value closer to the optimum in optimization functions than several well-known algorithms in the literature. Therefore, we can conclude that the different parts of the hypothesis exposed in Introduction Section have been accomplished. On the one hand, the proposal solution exposed in this dissertation can achieve a good performance in predict the different types of congestion that can appear in freeways and help to prevent traffic jams. With this, the first part of the hypothesis is covered. On the other hand, the second part of the hypothesis is concluded with the performance obtained by GACE in optimization function task, improving the results reached using well-known literature algorithms as Differential Evolution. Also, values for the different parameters have been established for future problems focusing in the dimension of these.

But, as it can be seen, the algorithm can still improve its performance in different themes. If we focus in general aspects of the algorithm, improvements to do can be:

- ◇ Take into account the parameters as values to optimize along the execution of the algorithm. This issue converts the problem in a multiobjetive problem: it is necessary to optimize not only the fitness function, but the different values of the parameters needed for improving the performance.

- ◇ Diversity in the population. Due to it is possible to find the same individuals in different sub-populations, sometimes the diversity of the general population gets lost along the process. This problem can be fixed, for example, increasing the mutation probability along the generations in the GA part, or making a reboot of the individuals in the CE part for updating the mean and deviation vectors.

In case of the congestion forecasting, main theme of this dissertation and where the proposal has been applied with more encourage, due to its importance in real-world problems, the algorithm and the design of the hierarchy can be improved in the follow aspects:

- ◇ Improvements in the results obtained for Light and Moderate congestion. As it can be seen in Experimentation chapter, some configurations of the algorithm have problems for make a good forecast in these types of congestion. This could be for the similarities between both values calculation. In future works, the differences between the calculus of the congestion values can be change in order to improve their correct classification.

- ◇ Use another kind of hierarchy. Serial hierarchy for congestion forecasting has been used in previous work to this dissertation. In our case, the parallel hierarchy has been applied and it has improved the results obtained in other literature works, not only for the use of this kind of hierarchy, but the use of the hybrid proposal to optimize its parts. The use of a hybrid hierarchy can be an interested way to act in order to see the effects of a different way to optimize the use of FRBSs.

- ◇ Data from other sources. The different datasets used in this dissertation and the associated paper is release to anyone who wants to use it. There are not many traffic datasets that can be obtained for free. Even so, governmental

platforms as the used one (PeMS) allow the researches to work with the data they collect. The use of other datasets can certify the good performance of the proposed technique, and it can improve the quality and the different types of traffic datasets.

- ◇ Time horizons. In this dissertation, short-term traffic congestion has been forecasted. The time horizons used have been 5, 15 and 30 minutes. Increase the number of time horizons, for example, 5 to 5 between 5 minutes and 60 minutes can specify when the congestion can be forecast with a bigger precision.

In function optimization, although it is not the principal task of the proposal technique, it has obtained good results in different kind of functions, some of them with a high complexity. Also, it has improved the results obtained by well-known literature algorithms as Differential Evolution. The possible action lines in this theme can be:

- ◇ In the experimentation, several values which the technique obtains good performance in high dimensions have been used in order to have better results in the dimensions where other techniques get worse and maintain a certain level of results in low dimensions. The goal now is to adapt these parameters to the dimension, and not have to choose only some of them, as well as we did with the population size.

- ◇ The use of other types of functions, as well as improve the performance adding or adapting some characteristics of the hybrid method to the functions and not only the operators can improve the performance and obtain a richer experimentation.

One of the principal characteristics of the proposed technique is its adaptation ability to other problems changing its operators and codification of the chromosome. Thanks to this, the amount of problems the technique can afford is huge. One of the problems where GACE can be applied is classification using ensembles. Classifier ensembles are sets of individual classifiers that offer a different approach to this problem using clustering for the search space and applying the best individual

classifiers to each one of the part of it. In this issue, GACE can be used to improve the different centroids and weights of the areas and the classifiers respectively in order to get a better result with the chosen classifiers of each area.

Another kind of issue, totally different from the exposed in this dissertation, is RFID systems and its tag collision problems. Summarizing, due to the shared nature of the wireless channel used by tags, these systems are prone to transmission collisions, which occurs when two or more tags transmit different information simultaneously. Fuzzy Logic has been applied to this problem in [Arjona 16]. Due to this, it is a good and different theme where GACE can help in order to find the correct values for the different parameters of Fuzzy Logic to get the best performance possible in avoid tag collisions.

There are always problems to solve. Real world problems such as congestion forecasting, pollution forecasting, route problems are current problems in many different projects, as can be found in Horizon 2020 calls, because of their importance in everyday life. Due to that, one of the most important things to do as a researcher is to find possible solutions to these problems. For this purpose, the improvement and development of new algorithms that can deal with large amount of data, and help to take decisions about important tasks like those mentioned before, are possible ways to present a solution. The proposed method in this dissertation tries to catch these concerns, and improves the results given by other techniques. On the other hand, the development of the proposal demonstrates that not everything is created, and for every problem, there are one or many solutions. Some of them are better than others. Otherwise, other solutions have not been thought yet. This dissertation made a proof of that.

6

Apéndice

6.1 Fórmulas de las funciones utilizadas

Cada tabla muestra las funciones utilizadas agrupadas por su tipo, y su forma matemática de manera que esto pueda ayudar a su entendimiento. En las definiciones de las funciones, se nombra un vector **z** que sustituye al argumento *x*. Este vector es un vector de variable transformada que tiene su óptimo en $\mathbf{z}^{opt} = 0$, si no se indica de otra manera. Para el resto de variables que aparecen en algunas funciones, como son s_i, T_{osz}, ó T_{asy}, su valor se muestra en la documentación que el lector puede encontrar en el siguiente link[1].

[1] http://coco.gforge.inria.fr/lib/exe/fetch.php?media=download3.6:bbobdocfunctions.pdf

Nombre	Fórmula
Sphere	$F_1(x) = x - x^{opt} + F_{opt}$
Ellipsoidal	$F_2(x) = \sum_{i=1}^{D} 10^{6\frac{i-1}{D-1}} z_i^2 + F_{opt}$
Rastrigin	$F_3(x) = 10\left(D - \sum_{i=1}^{D} \cos(2\pi z_i)\right) + \|\|\mathbf{z}\|\|^2 + F_{opt}$
Büche-Rastrigin	$F_4(x) = 10\left(D - \sum i = 1^D \cos 2\pi z_i\right) + \sum i = 1^D + 100 f_{pen} x + F_{opt}$
Linear Slope	$F_5(x) = \sum i = 1^D 5\|s_i\| - s_i z_i + F_{opt}$

Tabla 6.1: Funciones Separables: Nombre y fórmulas.

Nombre	Fórmula
Attractive Sector	$F_6(x) = T_{osz}\left(\sum i = 1^D (s_i z_i)^2\right)^{0.9} + F_{opt}$
Step Ellipsoidal	$F_7(x) = 0.1 max\left(\frac{\|z_1\|}{10^4}, \sum i = 1^D 10^{2\frac{i-1}{D-1}} z_i^2\right) + f_{pen}(x) + F_{opt}$
Rosenbrock , original	$F_8(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + F_{opt}$
Rosenbrock , rotated	$F_9(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + F_{opt}$

Tabla 6.2: Funciones con condicionamiento bajo o moderado: Nombres y funciones.

Nombre	Fórmula
Ellipsoidal	$F_{10}(x) = \sum_{i=1}^{D} 10^{6\frac{i-1}{D?1}} z_i^2 + F_{opt}$
Discus	$F_{11}(x) = 10^6 z_1^2 + \sum_{i=2}^{D} z_i^2 + F_{opt}$
Bent Cigar	$F_{12}(x) = z_1^2 + 10^6 \sum_{i=2}^{D} z_i^2 + F_{opt}$
Sharp Ridge	$F_{13}(x) = z_1^2 + 100\sqrt{\sum_{i=2}^{D} z_i^2} + F_{opt}$
Different Powers	$F_{14}(x) = \sqrt{\sum_{i=1}^{D} \|z_i\|^{2+4\frac{i-1}{D-1}}} + F_{opt}$

Tabla 6.3: Funciones con condicionamiento elevado y unimodales: Nombres y fórmulas.

Nombre	Fórmula
Rastrigin	$F_{15}(x) = 10\left(D - \sum_{i=1}^{D}\cos(2\pi z_i)\right) + \|\|\mathbf{z}\|\|^2 + F_{opt}$
Weirstrass	$F_{16}(x) = 10\left(\frac{1}{D}\sum_{i=1}^{D}\sum_{k=0}^{11}\frac{1}{2}^k\cos(2\pi 3^k(z_i+\frac{1}{2})) - f_0\right)^3 + \frac{10}{D}f_{pen}(x) + F_{opt}$
Schaffers F7	$F_{17}(x) = \left(\frac{1}{D-1}\sum_{i=1}^{D-1}\sqrt{s_i} + \sqrt{s_i}\sin^2(50s_i^{1/5})\right)^2 + 10f_{pen}(x) + F_{opt}$
Schaffers F7, moderately ill-conditioned	$F_{18}(x) = \left(\frac{1}{D-1}\sum_{i=1}^{D-1}\sqrt{s_i} + \sqrt{s_i}\sin^2(50s_i^{1/5})\right)^2 + 10f_{pen}(x) + F_{opt}$
Composite Griewank-Rosenbrock	$F_{19}(x) = \frac{10}{D-1}\sum_{i=1}^{D-1}\left(\frac{s_i}{4000} - \cos s_i\right) + 10 + F_{opt}$

Tabla 6.4: Funciones multimodales con estructura global adecuada: Nombres y fórmulas.

Nombre	Fórmula
Schwefel	$F_{20}(x) = -\frac{1}{D}\sum_{i=1}^{D} z_i\sin(\sqrt{\|z_i\|}) + 4.189828872724339 + 100f_{pen}(\frac{\mathbf{z}}{100}) + F_{opt}$
Gallagher's Gaussian 101-me Peaks	$F_{21}(x) = T_{osz}\left(10 - \text{máx}_{i=1}^{101} w_i\exp\left(-\frac{1}{2D}(x-y_i)^T R^T C_i R(x-y_i)\right)\right)^2 + f_{pen}(x) + F_{opt}$
Gallagher's Gaussian 21-hi Peaks	$F_{22}(x) = T_{osz}\left(10 - \text{máx}_{i=1}^{21} w_i\exp\left(-\frac{1}{2D}(x-y_i)^T R^T C_i R(x-y_i)\right)\right)^2 + f_{pen}(x) + F_{opt}$
Katsuura	$F_{23}(x) = \frac{10}{D^2}\prod_{i=1}^{D}\left(1 + i\sum_{j=1}^{32}\frac{\|2^j z_i - [2^j z_i]\|}{2^j}\right)^{\frac{10}{D^{1.2}}} - \frac{10}{D^2} + f_{pen}(x)$
Lunacek bi-Rastrigin	$F_{24}(x) = \text{mín}\left(\sum_{i=1}^{D}(\hat{x}_i - \mu_0)^2, dD + s\sum_{i=1}^{D}(\hat{x}_i - \mu_1)^2\right) + 10\left(D - \sum_{i=1}^{D}\cos(2\pi z_i)\right) + 10^4 + f_{pen}(x)$

Tabla 6.5: Funciones multimodales con estructura global débil: Nombres y fórmulas.

Bibliografía

[Abd-El-Wahed 11] W.F. Abd-El-Wahed, A.A. Mousa and M.A. El-Shorbagy. *Integrating particle swarm optimization with genetic algorithms for solving nonlinear optimization problems*. Journal of Computational and Applied Mathematics, Vol. 235, No. 5, pp. 1446–1453, 2011 *(pág. 20).*

[Adeli 11] M. Adeli and H. Zarabadipour. *Automatic disease diagnosis systems using pattern recognition based genetic algorithm and neural networks*. International Journal of Physical Sciences, Vol. 6, No. 25, pp. 6076–6081, 2011 *(pág. 16).*

[Ahmed 79] Mohamed S Ahmed and Allen R Cook. *Analysis of freeway traffic time-series data by using Box-Jenkins techniques*. Transportation Research Record, No. 722, 1979 *(pág. 5).*

[Aja-Fernández 08] S. Aja-Fernández and C. Alberola-López. *Matrix modeling of hierarchical fuzzy systems*. IEEE Transactions on Fuzzy Systems, Vol. 16, No. 3, pp. 585–599, 2008. cited By (since 1996)12 *(pág. 45).*

[Alba 05] Enrique Alba. Parallel metaheuristics: a new class of algorithms, volume 47. John Wiley & Sons, 2005 *(pág. 39).*

[Alba 09] E. Alba and B. Dorronsoro. Cellular genetic algorithms, volume 42. Springer-Verlag, 2009 *(pág. 91).*

[Alcalá 07] R. Alcalá, J. Alcalá-Fdez and F. Herrera. *A proposal for the genetic lateral tuning of linguistic fuzzy systems and its interaction with rule selection*. IEEE Transactions on Fuzzy Systems, Vol. 15, No. 4, pp. 616–635, 2007 *(pág. 60).*

[Arjona 16] L. Arjona, H. Landaluce, A. Perallos and E. Onieva. *Fast fuzzy anti-collision protocol for the RFID standard EPC Gen-2*. IET Electronic Letters, 2016. (Accepted for publication) *(pág. 105).*

[Awan 11] M.S.K. Awan and M.M. Awais. *Predicting weather events using fuzzy rule based system*. Applied Soft Computing Journal, Vol. 11, No. 1, pp. 56–63, 2011 *(pág. 23).*

[Azar 14] A.T. Azar and S.A. El-Said. *Performance analysis of support vector machines classifiers in breast cancer mammography recognition*. Neural Computing and Applications, Vol. 24, No. 5, pp. 1163–1177, 2014 *(pág. 30).*

[Bauza 13] R. Bauza and J. Gozalvez. *Traffic congestion detection in large-scale scenarios using vehicle-to-vehicle communications*. Journal of Network and Computer Applications, Vol. 36, No. 5, pp. 1295–1307, 2013 *(pág. 6).*

[Benitez 13] A.D. Benitez and J. Casillas. *Multi-objective genetic learning of serial hierarchical fuzzy systems for large-scale problems*. Soft Computing, Vol. 17, No. 1, pp. 165–194, 2013 *(pág. 42, 45).*

[Bertsimas 97] D. Bertsimas and J. N. Tsitsiklis. Introduction to linear optimization, volume 6. Athena Scientific, Belmont, MA, 1997 *(pág. 7, 28).*

[Bevilacqua 12] Vitoantonio Bevilacqua, Nicola Costantino, Mariagrazia Dotoli, Marco Falagario and Fabio Sciancalepore. *Strategic design and multi-objective optimisation of distribution networks*

based on genetic algorithms. International Journal of Computer Integrated Manufacturing, Vol. 25, No. 12, pp. 1139–1150, 2012 *(pág. 16).*

[Blum 11] C. Blum, J. Puchinger, G.R. Raidl and A. Roli. *Hybrid metaheuristics in combinatorial optimization: A survey*. Applied Soft Computing Journal, Vol. 11, No. 6, pp. 4135–4151, 2011 *(pág. 20).*

[Bouras 13] A. Bouras and W.P. Syam. *Hybrid chaos optimization and affine scaling search algorithm for solving linear programming problems*. Applied Soft Computing Journal, Vol. 13, No. 5, pp. 2703–2710, 2013 *(pág. 29).*

[Box 13] George EP Box, Gwilym M Jenkins and Gregory C Reinsel. Time series analysis: forecasting and control. John Wiley & Sons, 2013 *(pág. 5).*

[Cai 10] W. Cai and L. Ma. *Applications of critical temperature in minimizing functions of continuous variables with simulated annealing algorithm*. Computer Physics Communications, Vol. 181, No. 1, pp. 11–16, 2010 *(pág. 15).*

[Castro-Neto 09] Manoel Castro-Neto, Young-Seon Jeong, Myong-Kee Jeong and Lee D Han. *Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions*. Expert systems with applications, Vol. 36, No. 3, pp. 6164–6173, 2009 *(pág. 6).*

[Chen 04] Y. Chen, J. Dong and B. Yang. *Automatic design of hierarchical TS-FS model using Ant Programming and PSO algorithm*. volume 3192, pp. 285–294, 2004. cited By (since 1996)10 *(pág. 45).*

[Chen 07] Y. Chen, B. Yang, A. Abraham and L. Peng. *Automatic design of hierarchical Takagi-Sugeno type fuzzy systems using evolutionary algorithms*. IEEE Transactions on Fuzzy Systems, Vol. 15, No. 3, pp. 385–397, 2007. cited By (since 1996)57 *(pág. 45).*

[Chen 12] C.-F. Chen, M.-C. Lai and C.-C. Yeh. *Forecasting tourism demand based on empirical mode decomposition and neural network*. Knowledge-Based Systems, Vol. 26, pp. 281–287, 2012 *(pág. 6).*

[Chen 14] Z. Chen and C. Wang. *Research on continuous ant colony optimization algorithm and application in neural network modeling*. Journal of Multiple-Valued Logic and Soft Computing, Vol. 22, No. 3, pp. 317–340, 2014 *(pág. 31).*

[Ciornei 12] I. Ciornei and E. Kyriakides. *Hybrid ant colony-genetic algorithm (GAAPI) for global continuous optimization*. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, Vol. 42, No. 1, pp. 234–245, 2012 *(pág. 15).*

[Commission 14] European Commission. *Special Eurobarometer 422a, Quality of Transport*, 2014 *(pág. 3).*

[Cordón 01a] Oscar Cordón and Francisco Herrera. *Hybridizing genetic algorithms with sharing scheme and evolution strategies for designing approximate fuzzy rule-based systems*. Fuzzy sets and systems, Vol. 118, No. 2, pp. 235–255, 2001 *(pág. 17).*

[Cordón 01b] Oscar Cordón, Francisco Herrera, F Gomide, Frank Hoffmann and Luis Magdalena. *Ten years of genetic fuzzy systems: current framework and new trends*. In IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th, volume 3, pp. 1241–1246. IEEE, 2001 *(pág. 17).*

[Cotta-Porras 98] Carlos Cotta-Porras. *Study of hybridization techniques and their application to the design of evolutionary algorithms*. AI Communications, Vol. 11, No. 3, pp. 223–224, 1998 *(pág. 39).*

[De Boer 05] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor and Reuven Y Rubinstein. *A tutorial on the cross-entropy method*. Annals of operations research, Vol. 134, No. 1, pp. 19–67, 2005 *(pág. 19).*

[De Jong 75] Kenneth Alan De Jong. *Analysis of the behavior of a class of genetic adaptive systems*. 1975 *(pág. 32, 33).*

[De Ridder 13] K. De Ridder, U. Kumar, D. Lauwaet, S. Van Looy and W. Lefebvre. *Kalman Filter-Based Air Quality Forecast Adjustment*. NATO Science for Peace and Security Series C: Environmental Security, Vol. 137, pp. 177–181, 2013 *(pág. 6).*

[Deb 02] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal and TAMT Meyarivan. *A fast and elitist multiobjective genetic algorithm: NSGA-II*. Evolutionary Computation, IEEE Transactions on, Vol. 6, No. 2, pp. 182–197, 2002 *(pág. 30).*

[Deep 10] Kusum Deep and Hadush Mebrahtu Adane. *New Variations of Order Crossover for Travelling Salesman Problem*. International Journal of Combinatorial Optimization Problems and Informatics, Vol. 2, No. 1, pp. 2–13, 2010 *(pág. 63).*

[DellÁmico 15] M. DellÁmico, S. Falavigna and M. Iori. *Optimization of a real-world auto-carrier transportation problem*. Transportation Science, Vol. 49, No. 2, pp. 402–419, 2015 *(pág. 7).*

[Derrac 11] J. Derrac, S. García, D. Molina and F. Herrera. *A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms*. Swarm and Evolutionary Computation, Vol. 1, No. 1, pp. 3–18, 2011 *(pág. 96).*

[Dorigo 97] M. Dorigo and L.M. Gambardella. *Ant colony system: A cooperative learning approach to the traveling salesman problem*. IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1, pp. 53–66, 1997 *(pág. 15).*

[Eiben 97] Agoston E Eiben and Thomas Bäck. *Empirical investigation of multiparent recombination operators in evolution strategies*. Evolutionary Computation, Vol. 5, No. 3, pp. 347–365, 1997 *(pág. 32, 33).*

[Eiselt 87] H. A. Eiselt, G. Pederzoli and C. Sandblom. Continuous optimization models, volume 1. Walter De Gruyter, 1987 *(pág. 7, 28).*

[Elsheikh 14] A. Elsheikh. *Derivative-based hybrid heuristics for continuous-time simulation optimization*. Simulation Modelling Practice and Theory, Vol. 46, pp. 164–175, 2014 *(pág. 29).*

[Eshelman 93] Larry J Eshelman. *Real-coded Genetic Algorithms and Interval-Schemata*. Foundations of genetic algorithms, Vol. 2, pp. 187–202, 1993 *(pág. 64).*

[Fernández 09] Alberto Fernández, María José del Jesus and Francisco Herrera. *Hierarchical fuzzy rule based classification systems with genetic rule selection for imbalanced data-sets*. International Journal of Approximate Reasoning, Vol. 50, No. 3, pp. 561–577, 2009 *(pág. 42).*

[Finner 93] H Finner. *On a monotonicity problem in step-down multiple test procedures*. Journal of the American Statistical Association, Vol. 88, No. 423, pp. 920–923, 1993 *(pág. 96).*

[Garcia-Villoria 10] A. Garcia-Villoria, A. Corominas and R. Pastor. *Solving the response time variability problem by means of the cross-entropy method*. International Journal of Manufacturing Tech-

nology and Management, Vol. 20, No. 1-4, pp. 316–330, 2010 *(pág. 18).*

[Ghorbani 10] Reza Ghorbani, Eric Bibeau and Shaahin Filizadeh. *On conversion of hybrid electric vehicles to plug-in*. IEEE Transactions on Vehicular Technology, Vol. 59, No. 4, pp. 2016–2020, 2010 *(pág. 23).*

[Goldberg 91] David E Goldberg and Kalyanmoy Deb. *A comparative analysis of selection schemes used in genetic algorithms*. Urbana, Vol. 51, pp. 61801–2996, 1991 *(pág. 62).*

[Gu 11] Y. Gu, Y. Liu and J. Xu. *Improved hybrid intelligent method for urban road traffic flow forecasting based on chaos-PSO optimization*. International Journal of Advancements in Computing Technology, Vol. 3, No. 7, pp. 282–290, 2011 *(pág. 6).*

[Gu 12] Y. Gu, P. Qu and Z. Feng. *Urban traffic flow prediction based on FNN and improved GA optimization*. International Review on Computers and Software, Vol. 7, No. 3, pp. 1316–1319, 2012 *(pág. 6).*

[Haber 10] R.E. Haber, R.M. Del Toro and A. Gajate. *Optimal fuzzy control system using the cross-entropy method. A case study of a drilling process*. Information Sciences, Vol. 180, No. 14, pp. 2777–2792, 2010 *(pág. 18).*

[Hansen 01] N. Hansen and A. Ostermeier. *Completely derandomized self-adaptation in evolution strategies*. Evolutionary Computation, Vol. 9, No. 2, pp. 159–195, 2001 *(pág. 7, 15).*

[He 13] K. He, Y. Jin and W. Huang. *Heuristics for two-dimensional strip packing problem with 90 rotations*. Expert Systems with Applications, Vol. 40, No. 14, pp. 5542–5550, 2013 *(pág. 29).*

[Hernández 13] S.A. Hernández, G. Leguizamón and E. Mezura-Montes. *Hybridization of Differential Evolution using Hill Climbing to solve Constrained Optimization Problems*. Revista Iberoamericana de Inteligencia Artificial, Vol. 16, No. 52, pp. 3–15, 2013 *(pág. 20).*

[Herrera 98] Francisco Herrera, Manuel Lozano and Jose L. Verdegay. *Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis*. Artificial intelligence review, Vol. 12, No. 4, pp. 265–319, 1998 *(pág. 64).*

[Holland 75] J. H. Holland. Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. Univ. of Michigan Press, 1975 *(pág. 15, 16, 29).*

[Holm 79] S. Holm. *A simple sequentially rejective multiple test procedure*. Scandinavian Journal of Statistics, pp. 65–70, 1979 *(pág. 96).*

[Holtschulte 13] N. J. Holtschulte and M. Moses. *Benchmarking cellular genetic algorithms on the BBOB noiseless testbed*. In Proceedings of the Companion Publication of the 2013 Conference on Genetic and Evolutionary Computation, pp. 1201–1208. ACM, 2013 *(pág. 91).*

[Hyndman 06] R.J. Hyndman and A.B. Koehler. *Another look at measures of forecast accuracy*. International Journal of Forecasting, Vol. 22, No. 4, pp. 679–688, 2006 *(pág. 69).*

[Iglesias 10] José Antonio Iglesias, Plamen Angelov, Agapito Ledezma and Araceli Sanchis. *Human activity recognition based on evolving fuzzy systems*. International Journal of Neural Systems, Vol. 20, No. 05, pp. 355–364, 2010 *(pág. 23).*

[Jacobson 04] S. H. et al. Jacobson. *Global optimization performance measures for generalized hill climbing algorithms*. Journal of Global Optimization, Vol. 29, No. 2, pp. 173–190, 2004 *(pág. 91).*

[Jelleli 10] T. Jelleli and A. Alimi. *Automatic design of a least complicated hierarchical fuzzy system*. In: 6th IEEE World Congress on computational intelligence, pp. 1–7, 2010. cited By (since 1996)1 *(pág. 45).*

[Jia 06] Lei Jia, Licai Yang, Qingjie Kong and Shu Lin. *Study of artificial immune clustering algorithm and its applications to urban traffic control*. International Journal of Information Technology, Vol. 12, No. 3, pp. 1–6, 2006 *(pág. 6).*

[Joo 09] M.G. Joo and T. Sudkamp. *A method of converting a fuzzy system to a two-layered hierarchical fuzzy system and its runtime efficiency*. IEEE Transactions on Fuzzy Systems, Vol. 17, No. 1, pp. 93–103, 2009. cited By (since 1996)10 *(pág. 45).*

[Jourdan 09] L. Jourdan, M. Basseur and E.-G. Talbi. *Hybridizing exact methods and metaheuristics: A taxonomy*. European Journal of Operational Research, Vol. 199, No. 3, pp. 620–629, 2009 *(pág. 29, 39).*

[Keller 85] James M Keller, Michael R Gray and James A Givens. *A fuzzy k-nearest neighbor algorithm*. Systems, Man and Cybernetics, IEEE Transactions on, No. 4, pp. 580–585, 1985 *(pág. 58).*

[Kennedy 10] J. Kennedy. *Particle swarm optimization*. In Encyclopedia of Machine Learning, pp. 760–766. Springer-Verlag, 2010 *(pág. 15).*

[Kullback 51] S. Kullback and R. A. Leibler. *On information and sufficiency*. The annals of mathematical statistics, Vol. 22, No. 1, pp. 79–86, 1951 *(pág. 18).*

[Kuo 15] R.J. Kuo, Y.H. Lee, F.E. Zulvia and F.C. Tien. *Solving bi-level linear programming problem through hybrid of immune genetic algorithm and particle swarm optimization algorithm*. Applied Mathematics and Computation, Vol. 266, pp. 1013–1026, 2015 *(pág. 29).*

[Lagarias 98] J.C. Lagarias, J.A. Reeds, M.H. Wright and P.E. Wright. *Convergence properties of the Nelder-Mead simplex method in low dimensions*. SIAM Journal on Optimization, Vol. 9, No. 1, pp. 112–147, 1998 *(pág. 29).*

[Larranaga 99] Pedro Larranaga, Cindy M. H. Kuijpers, Roberto H. Murga, Inaki Inza and Sejla Dizdarevic. *Genetic algorithms for the travelling salesman problem: A review of representations and operators*. Artificial Intelligence Review, Vol. 13, No. 2, pp. 129–170, 1999 *(pág. 64).*

[Li 11] Ruimin Li and Geoffrey Rose. *Incorporating uncertainty into short-term travel time predictions*. Transportation Research Part C: Emerging Technologies, Vol. 19, No. 6, pp. 1006–1018, 2011 *(pág. 6).*

[Liao 14] T. Liao, T. Stützle, M.A. Montes De Oca and M. Dorigo. *A unified ant colony optimization algorithm for continuous optimization*. European Journal of Operational Research, Vol. 234, No. 3, pp. 597–609, 2014 *(pág. 31).*

[Liu 12] H. Liu, H.-Q. Tian and Y.-F. Li. *Comparison of two new ARIMA-ANN and ARIMA-Kalman hybrid methods for wind speed prediction*. Applied Energy, Vol. 98, pp. 415–424, 2012 *(pág. 6).*

[Lopez-Garcia 15] Pedro Lopez-Garcia, Enrique Onieva, Eneko Osaba, Antonio D. Masegosa and Asier Perallos. *Hybridizing Genetic Algorithm with Cross Entropy for Solving Continuous Functions*. In Proceedings of the Companion Publication of the

2015 Conference on Genetic and Evolutionary Computation, GECCO Companion '15, pp. 763–764, 2015 *(pág. 49).*

[Lopez-Garcia 16a] P. Lopez-Garcia, E. Onieva, E. Osaba, A. D. Masegosa and A. Perallos. *A Hybrid Method for Short-Term Traffic Congestion Forecasting Using Genetic Algorithms and Cross Entropy*. IEEE Transactions on Intelligent Transportation Systems, Vol. 17, No. 2, pp. 557–569, 2016 *(pág. 7, 23, 45, 47).*

[Lopez-Garcia 16b] P. Lopez-Garcia, E. Onieva, E. Osaba, A.D. Masegosa and A. Perallos. *GACE: A meta-heuristic based in the hybridization of Genetic Algorithms and Cross Entropy methods for continuous optimization*. Expert Systems with Applications, Vol. 55, pp. 508–519, 2016 *(pág. 49).*

[López 13] Victoria López, Alberto Fernández, Salvador García, Vasile Palade and Francisco Herrera. *An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics*. Information Sciences, Vol. 250, pp. 113–141, 2013 *(pág. 58).*

[Lozano 11] Manuel Lozano, Daniel Molina and Francisco Herrera. *Editorial scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems*. Soft Computing, Vol. 15, No. 11, pp. 2085–2087, 2011 *(pág. 32, 33).*

[Lu 14] Y. Lu, Y. Liu, C. Gao, L. Tao and Z. Zhang. *A novel Physarum-Based ant colony system for solving the real-world traveling salesman problem*. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 8794, pp. 173–180, 2014 *(pág. 7).*

[Mandal 11] A. Mandal, A.K. Das, P. Mukherjee, S. Das and P.N. Suganthan. *Modified differential evolution with local search algorithm for real world optimization*. pp. 1565–1572, 2011 *(pág. 20).*

[Mansoori 08] Eghbal G Mansoori, Mansoor J Zolghadri and Seraj D Katebi. *SGERD: A steady-state genetic algorithm for extracting fuzzy classification rules from data*. IEEE Transactions on Fuzzy Systems, Vol. 16, No. 4, pp. 1061–1071, 2008 *(pág. 71).*

[Ömer Faruk 10] D. Ömer Faruk. *A hybrid neural network and ARIMA model for water quality time series prediction*. Engineering Applications of Artificial Intelligence, Vol. 23, No. 4, pp. 586–594, 2010 *(pág. 6).*

[Muñoz 13] Mario A Muñoz, Michael Kirley and Saman K Halgamuge. *The algorithm selection problem on the continuous optimization domain*. In Computational Intelligence in Intelligent Data Analysis, pp. 75–89. Springer, 2013 *(pág. 36).*

[Mühlenbein 93] Heinz Mühlenbein and Dirk Schlierkamp-Voosen. *Predictive models for the breeder genetic algorithm i. continuous parameter optimization*. Evolutionary computation, Vol. 1, No. 1, pp. 25–49, 1993 *(pág. 64).*

[Murat 14] Y.S. Murat, S. Kutluhan and N. Uludag. *Use of fuzzy optimization and linear goal programming approaches in urban bus lines organization*. Advances in Intelligent Systems and Computing, Vol. 223, pp. 377–387, 2014 *(pág. 29).*

[Neri 10] F. Neri and V. Tirronen. *Recent advances in differential evolution: A survey and experimental analysis*. Artificial Intelligence Review, Vol. 33, No. 1-2, pp. 61–106, 2010 *(pág. 15).*

[Nocedal 06] J. Nocedal and S. J. Wright. Numerical optimization. Springer, 2006 *(pág. 28).*

[Okulewicz 13] M. Okulewicz and J. Mańdziuk. *Application of particle swarm optimization algorithm to dynamic vehicle routing problem*. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 7895 LNAI, No. PART 2, pp. 547–558, 2013 *(pág. 31).*

[Okutani 84] Iwao Okutani and Yorgos J Stephanedes. *Dynamic prediction of traffic volume through Kalman filtering theory*. Transportation Research Part B: Methodological, Vol. 18, No. 1, pp. 1–11, 1984 *(pág. 5).*

[Onieva 09] E. Onieva, J. Alonso, J. Perez, V. Milanes and T. de Pedro. *Autonomous car fuzzy control modeled by iterative genetic algorithms*. In FUZZ-IEEE 2009. IEEE International Conference on, pp. 1615–1620, Aug 2009 *(pág. 23).*

[Onieva 12] Enrique Onieva, Vicente Milanés, Jorge Villagra, Joshué Pérez and Jorge Godoy. *Genetic optimization of a vehicle fuzzy decision system for intersections*. Expert Systems with Applications, Vol. 39, No. 18, pp. 13148–13157, 2012 *(pág. 17).*

[Osaba 13] Eneko Osaba, Enrique Onieva, Roberto Carballedo, Fernando Diaz, Asier Perallos and Xiao Zhang. *A multi-crossover and adaptive island based population algorithm for solving routing problems*. Journal of Zhejiang University SCIENCE C, Vol. 14, No. 11, pp. 815–821, 2013 *(pág. 16).*

[Osaba 14] Eneko Osaba, Fernando Diaz and Enrique Onieva. *Golden ball: a novel meta-heuristic to solve combinatorial optimization problems based on soccer concepts*. Applied Intelligence, pp. 1–22, 2014 *(pág. 16, 29, 31).*

[Papadimitriou 98] C. H. Papadimitriou and K. Steiglitz. Combinatorial optimization: Algorithms and complexity. Courier Corporation, 1998 *(pág. 7, 28).*

[Pošík 12] Petr Pošík and Václav Klemš. *Benchmarking the differential evolution with adaptive encoding on noiseless functions*. In Proceedings of the 14th annual conference companion on Genetic and evolutionary computation, pp. 189–196. ACM, 2012 *(pág. 90).*

[Purwar 15] A. Purwar and S.K. Singh. *Hybrid prediction model with missing value imputation for medical data*. Expert Systems with Applications, Vol. 42, No. 13, pp. 5621–5631, 2015 *(pág. 20).*

[Qiao 11] J. Qiao, N. Yang and J. Gao. *Two-stage fuzzy logic controller for signalized intersection*. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, Vol. 41, No. 1, pp. 178–184, 2011 *(pág. 16).*

[Quinlan 14] J Ross Quinlan. C4. 5: programs for machine learning. Elsevier, 2014 *(pág. 71).*

[Raidl 06] G.R. Raidl. *A unified view on hybrid metaheuristics*. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 4030 LNCS, pp. 1–12, 2006 *(pág. 39).*

[Rubinstein 97] R.Y. Rubinstein. *Optimization of computer simulation models with rare events*. European Journal of Operational Research, Vol. 99, No. 1, pp. 89–112, 1997 *(pág. 17).*

[Rubinstein 99] R. Rubinstein. *The cross-entropy method for combinatorial and continuous optimization*. Methodology and Computing in Applied Probability, Vol. 1, No. 2, pp. 127–190, 1999 *(pág. 7, 15, 18).*

[Rzeszotko 12] J. Rzeszotko and S.H. Nguyen. *Machine learning for traffic prediction*. Fundamenta Informaticae, Vol. 119, No. 3–4, pp. 407–420, 2012 *(pág. 6).*

[Saha 10] S. Saha, S. Moorthi, H.-L. Pan, X. Wu, J. Wang, S. Nadiga, P. Tripp, R. Kistler, J. Woollen, D. Behringer, H. Liu, D. Stokes, R. Grumbine, G. Gayno, J. Wang, Y.-T. Hou, H.-Y. Chuang, H.-M.H. Juang, J. Sela, M. Iredell, R. Treadon, D. Kleist, P. Van Delst, D. Keyser, J. Derber, M. Ek, J. Meng, H. Wei, R. Yang, S. Lord, H. Van Den Dool, A. Kumar, W. Wang, C. Long, M. Chelliah, Y. Xue, B. Huang, J.-K. Schemm, W. Ebisuzaki, R. Lin, P. Xie, M. Chen, S. Zhou, W. Higgins, C.-Z. Zou, Q. Liu, Y. Chen, Y. Han, L. Cucurull, R.W. Reynolds, G. Rutledge and M. Goldberg. *The NCEP climate forecast system reanalysis*. Bulletin of the American Meteorological Society, Vol. 91, No. 8, pp. 1015–1057, 2010 *(pág. 6).*

[Sangsawang 14] C. Sangsawang, K. Sethanan, T. Fujimoto and M. Gen. *Metaheuristics optimization approaches for two-stage reentrant flexible flow shop with blocking constraint*. Expert Systems with Applications, Vol. 42, No. 5, pp. 2395–2410, 2014 *(pág. 20).*

[Sawyerr 13] B. A. Sawyerr, A. O. Adewumi and M.M. Ali. *Benchmarking projection-based real coded genetic algorithm on BBOB-2013 noiseless function testbed*. In Proceedings of the Companion Publication of the 2013 Conference on Genetic and Evolutionary Computation, pp. 1193–1200, 2013 *(pág. 90).*

[Schwefel 81] H. Schwefel. Numerical optimization of computer models. John Wiley & Sons, 1981 *(pág. 7, 28).*

[Segura 15] Carlos Segura, Carlos A Coello Coello and Alfredo G Hernández-Díaz. *Improving the vector generation strategy*

of Differential Evolution for large-scale optimization. Information Sciences, Vol. 323, pp. 106–129, 2015 *(pág. 39).*

[Shimojima 95] K. Shimojima, T. Fukuda and Y. Hasegawa. *Self-tuning fuzzy modeling with adaptive membership function, rules, and hierarchical structure based on genetic algorithm*. Fuzzy Sets and Systems, Vol. 71, No. 3, pp. 295–309, 1995. cited By (since 1996)101 *(pág. 45).*

[Silva 14] E. Silva, J.F. Oliveira and G. Wäscher. *2DCPackGen: A problem generator for two-dimensional rectangular cutting and packing problems*. European Journal of Operational Research, Vol. 237, No. 3, pp. 846–856, 2014 *(pág. 29).*

[Skycomp 09] Cox Skycomp Inc. in association with Whytney Bailey and LLC Magnani. *Major Highway Performance Ratings and Bottleneck Inventory, State of Maryland - Spring 2008*. 2009 *(pág. 55).*

[Sousa 04] Tiago Sousa, Arlindo Silva and Ana Neves. *Particle swarm based data mining algorithms for classification tasks*. Parallel Computing, Vol. 30, No. 5, pp. 767–783, 2004 *(pág. 71).*

[Stathopoulos 08] A. Stathopoulos, L. Dimitriou and T. Tsekeris. *Fuzzy modeling approach for combined forecasting of urban traffic flow*. Computer-Aided Civil and Infrastructure Engineering, Vol. 23, No. 7, pp. 521–535, 2008 *(pág. 6).*

[Stellet 15] J.E. Stellet, F. Straub, J. Schumacher, W. Branz and J.M. Zollner. *Estimating the Process Noise Variance for Vehicle Motion Models*. pp. 1512–1519, 2015 *(pág. 7).*

[Syberfeldt 09] A. Syberfeldt, A. Ng, R.I. John and P. Moore. *Multi-objective evolutionary simulation-optimisation of a real-world manufacturing problem*. Robotics and Computer-Integrated Manufacturing, Vol. 25, No. 6, pp. 926–931, 2009 *(pág. 7).*

[Talbi 02] E-G Talbi. *A taxonomy of hybrid metaheuristics*. Journal of heuristics, Vol. 8, No. 5, pp. 541–564, 2002 *(pág. 36, 39).*

[Tan 07] M.-C. Tan, L.-B. Feng and J.-M. Xu. *Traffic flow prediction based on hybrid ARIMA and ANN model*. Zhongguo Gonglu Xuebao/China Journal of Highway and Transport, Vol. 20, No. 4, pp. 118–121, 2007 *(pág. 5, 6).*

[Taylor 06] J.W. Taylor, L.M. de Menezes and P.E. McSharry. *A comparison of univariate methods for forecasting electricity demand up to a day ahead*. International Journal of Forecasting, Vol. 22, No. 1, pp. 1–16, 2006 *(pág. 6).*

[Thakur 14] M. Thakur. *A new genetic algorithm for global optimization of multimodal continuous functions*. Journal of Computational Science, Vol. 5, No. 2, pp. 298–311, 2014 *(pág. 15, 16).*

[Toledo 14] C.F.M. Toledo, L. Oliveira and P.M. França. *Global optimization using a genetic algorithm with hierarchically structured population*. Journal of Computational and Applied Mathematics, Vol. 261, pp. 341–351, 2014 *(pág. 31).*

[Topcuoglu 07] H. R. Topcuoglu, B. Demiroz and M. Kandemir. *Solving the register allocation problem for embedded systems using a hybrid evolutionary algorithm*. IEEE Transactions on Evolutionary Computation, Vol. 11, No. 5, pp. 620–634, 2007 *(pág. 20).*

[Valizadeh 14] M. Valizadeh, S. Syafiie and I.S. Ahamad. *Multi-objective particle swarm optimization for optimal planning of biodiesel supply chain in Malaysia*. Advances in Soft Computing, Vol. 287, pp. 293–302, 2014 *(pág. 30).*

[Van Laarhoven 87] P. JM Van Laarhoven and E. HL Aarts. Simulated annealing. Springer-Verlag, 1987 *(pág. 15).*

[Verdegay 08] José L Verdegay, Ronald R Yager and Piero P Bonissone. *On heuristics as a fundamental constituent of soft computing*. Fuzzy Sets and Systems, Vol. 159, No. 7, pp. 846–855, 2008 *(pág. 13).*

[Wang 04] ZG Wang, YS Wong and M Rahman. *Optimisation of multi-pass milling using genetic algorithm and genetic simulated annealing*. The International Journal of Advanced Manufacturing Technology, Vol. 24, No. 9-10, pp. 727–732, 2004 *(pág. 36, 39).*

[Wang 05] Yibing Wang and Markos Papageorgiou. *Real-time freeway traffic state estimation based on extended Kalman filter: a general approach*. Transportation Research Part B: Methodological, Vol. 39, No. 2, pp. 141–167, 2005 *(pág. 6).*

[Wang 09] Y. Wang and Z. Cai. *A hybrid multi-swarm particle swarm optimization to solve constrained optimization problems*. Frontiers of Computer Science in China, Vol. 3, No. 1, pp. 38–52, 2009 *(pág. 7).*

[Wang 11] Y. Wang, B. Li, T. Weise, J. Wang, B. Yuan and Q. Tian. *Self-adaptive learning based particle swarm optimization*. Information Sciences, Vol. 181, No. 20, pp. 4515–4538, 2011 *(pág. 15).*

[Welch 95] Greg Welch and Gary Bishop. *An introduction to the Kalman filter*, 1995 *(pág. 5).*

[Whitley 95] D. Whitley, K. Mathias, S. Rana and J. Dzubera. *Building Better Test Functions*. In Proceedings of the Sixth International Conference on Genetic Algorithms, pp. 239–246. Morgan Kaufmann, 1995 *(pág. 32).*

[Williams 98] Billy M Williams, Priya K Durvasula and Donald E Brown. *Urban freeway traffic flow prediction: application of seasonal autoregressive integrated moving average and exponential smoothing models*. Transportation Research Record: Journal of the Transportation Research Board, Vol. 1644, No. 1, pp. 132–141, 1998 *(pág. 5).*

[Wolpert 97] David H Wolpert and William G Macready. *No free lunch theorems for optimization*. Evolutionary Computation, IEEE Transactions on, Vol. 1, No. 1, pp. 67–82, 1997 *(pág. 32).*

[Wright 99] S. J. Wright. *Continuous Optimization (Nonlinear and Linear Programming)*. Foundations of Computer-Aided Process Design, 1999 *(pág. 29).*

[Wright 05] M.H. Wright. *The interior-point revolution in optimization: History, recent developments, and lasting consequences*. Bulletin of the American Mathematical Society, Vol. 42, No. 1, pp. 39–56, 2005 *(pág. 29).*

[Xia 12] M. Xia and Z. Xu. *Entropy/cross entropy-based group decision making under intuitionistic fuzzy environment*. Information Fusion, Vol. 13, No. 1, pp. 31–47, 2012 *(pág. 18).*

[Xu 10] Y. Xu, Y. Wu, G. Wu, J. Xu, B. Liu and L. Sun. *Data collection for the detection of urban traffic congestion by VANETs*. Proceedings - 2010 IEEE Asia-Pacific Services Computing Conference, APSCC 2010, pp. 405–410, 2010 *(pág. 6).*

[Yang 07] J.-S. Yang. *Application of the Kalman filter to the arterial travel time prediction: A special event case study*. Control and Intelligent Systems, Vol. 35, No. 1, pp. 79–85, 2007 *(pág. 5).*

[Zadeh 65] Lotfi A Zadeh. *Fuzzy sets*. Information and control, Vol. 8, No. 3, pp. 338–353, 1965 *(pág. 21).*

[Zadeh 94] Lotfi A. Zadeh. *Soft computing and fuzzy logic*. IEEE Software, Vol. 11, No. 6, pp. 48–56, 1994 *(pág. 13).*

[Zadeh 96] Lotfi A Zadeh. *Fuzzy logic= computing with words*. Fuzzy Systems, IEEE Transactions on, Vol. 4, No. 2, pp. 103–111, 1996 *(pág. 21).*

[Zhang 12a] J. Zhang and R. Lv. *Fuzzy particle swarm optimization algorithm in solving traveling salesman problem*. International Review on Computers and Software, Vol. 7, No. 5, pp. 2593–2597, 2012 *(pág. 29).*

[Zhang 12b] L. Zhang, J. Ma and C. Zhu. *Theory modeling and application of an adaptive Kalman filter for short-term traffic flow prediction*. Journal of Information and Computational Science, Vol. 9, No. 16, pp. 5101–5109, 2012 *(pág. 5).*

[Zhang 14] Xiao Zhang, Enrique Onieva, Asier Perallos, Eneko Osaba and Victor Lee. *Hierarchical fuzzy rule-based system optimized with genetic algorithms for short term traffic congestion prediction*. Transportation Research Part C: Emerging Technologies, 2014 *(pág. 7, 45).*