Optimización de Sistemas Basados en Reglas Difusas para la predicción de congestión a corto plazo

P. López-García, E. Onieva, A. Perallos, L. Arjona, E. Osaba

Resumen— **El incremento de vehículos en las ciudades y autopistas conlleva una serie de problemas. Entre ellos, las aglomeraciones de tráfico es uno de los más importantes. Predecir la congestión a corto plazo en un punto o una sección de carretera se ha convertido en uno de los puntos esenciales de los Sistemas Inteligentes de Transporte. Este trabajo presenta un método híbrido de optimización de los elementos de una jerarquía de Sistemas Basados en Reglas Difusas, combinando un Algoritmo Genético y un método de Entropía Cruzada. Los sistemas se utilizan para predecir la congestión en un tramo o sección de carretera a corto plazo: 5, 15 y 30 minutos. Los resultados obtenidos mejoran la precisión de los métodos por separado así como de otros métodos clásicos de la literatura.**

Palabras clave—**Sistemas Basados en Reglas Difusas, Algoritmos Genéticos, Entropía Cruzada, Optimización, Sistemas Inteligentes de Transporte, Congestión de tráfico**

I. Introducción

Según la Federación Internacional de Carreteras, en la sociedad actual, el número de vehículos en ciudades y carreteras aumenta contínuamente. Existen muchos problemas relacionados con este incremento, siendo uno de los más importantes la congestión de las arterias principales de las ciudades. Derivados de este problema surgen otros, tales como el malgasto de tiempo y combustible, la disminución de la precisión en la predicción de los tiempos de viaje, el desgaste del vehículo, estrés y frustración de los conductores, y bloqueo de vehículos de emergencia, entre otros.

La temprana detección de la congestión del tráfico es un tema fundamental en el campo de los Sistemas Inteligentes de Transporte (SIT). Los SIT son el vínculo entre las tecnologías de la información y la comunicación y los vehículos y redes que transportan personas o mercancías. Realizar una predicción satisfactoria del estado del tráfico puede ayudar a tomar medidas para, por ejemplo, incrementar la eficacia y el rendimiento de los sistemas de transporte, el descenso de la polución, que conlleva un ahorro de energía, y ahorro en infraestructura pública para las instituciones.

Para obtener dicho objetivo, dos de los métodos más usados en la previsión del estado del tráfico en la última década han sido el filtro de Kalman [1] y ARIMA (Autoregressive Integrated Moving Average) [2]. Ambos son modelos regresivos que encuentran patrones para la predicción de un valor en el futuro.

En los últimos años, se han desarrollado otras alternativas, como las comunicaciones Vehículo a Vehículo [3] o el esquema Time-of-Day que divide un día en intervalos e intenta encontrar los controladores óptimos para cada uno [4].

Otra clase de alternativas son las técnicas de Soft Computing como las Máquinas Vector Soporte, Redes Neuronales, Algoritmos Genéticos (AG) o Sistemas Basados en Reglas Difusas (SBRD). Estas se han usado tanto por separado [5] ó combinando diferentes métodos [6].

Siguiendo con esto último, la falta en la literatura de métodos que combinen técnicas tan extendidas como son los AG con otras poco utilizadas en esta clase de problemas como la Entropía Cruzada (EC) [7] motiva a la realización de un método híbrido que sume las ventajas de ambas técnicas para la optimización de Sistemas Jerárquicos Basados en Reglas Difusas (SJBRD) con la intención de predecir con el menor error posible la congestión a corto plazo en una autopista. La hibridación de un AG con un EC proporciona tanto exploración como explotación de los individuos de la población.

Este documento se estructura como sigue. La Sección II contiene la información preliminar sobre los métodos utilizados en este trabajo. La Sección III presenta un estudio sobre la estructura del algoritmo. En la Sección IV se presenta la experimentación realizada mientras que la Sección V contiene las conclusiones aportadas por los resultados obtenidos y líneas de trabajo futuras.

II. Preliminares

En esta sección se explica de manera breve los diferentes métodos y técnicas que se aplican en este trabajo. Por tanto, la sección queda estructurada como sigue. En la Sección II-A se define la Soft Computing, Lógica Difusa y los Sistemas Basados en Reglas difusas. Por último, las Secciones II-B y II-C contienen una descripción breve sobre los algoritmos genéticos y el método de Entropía Cruzada.

A. Soft Computing y Lógica Difusa

Se puede definir la Soft Computing como un conjunto de metodologías cuyo objetivo es explotar la

Instituto Tecnológico Deusto (DeustoTech), Universidad de Deusto, Bilbao Mail: {p.lopez,enrique.onieva,perallos,laura.arjona,e.osaba@deusto.es}

tolerancia al error y la incertidumbre para conseguir manejabilidad, robustez y soluciones de bajo coste, y mejores representaciones de la realidad [8].

Podemos encontrarla definida más recientemente en [9], donde se dice que la Soft Computing engloba 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.

Una de las partes de la Soft Computing utilizada en este trabajo es la Lógica Difusa. Introducida por Zadeh en 1965 [10], permite procesar información imprecisa usando reglas del tipo SI-ENTONCES (IF-THEN).

En problemas de tráfico, como en [11], 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 Sistemas Basados en Reglas Difusas (SBRD). Estos sistemas están formados por varias entradas, un sistema de inferencia difuso con un conjunto de reglas en su interior y una salida. Los SBRDs son utilizados en diferentes tipos de problemas de la vida real [12].

Una variante de un SBRD básico es el SBRD jerárquico (SJBRD) [13], que cuenta con varios SBRDs, los cuáles están unidos unos a otros de tal forma que la salida de uno de ellos es la entrada de otro. Gracias a su estructura, estos sistemas 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 la *maldición de la dimensionalidad*, [14]. Dependiendo de la forma en la que estructuren los sistemas, existen tres modelos diferentes de SJBRD:

1. SJBRD en serie: la salida de un SBRD es la entrada del siguiente. También es posible incluir variables externas como una entrada del siguiente sistema.
2. SJBRD 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. SJBRD híbridos: Una combinación de los casos anteriores.

Este trabajo está enfocado en el segundo tipo de sistemas: SJBRD en paralelo (*Parallel HSBRD*, SJPBRD) con la restricción de que cada sistema únicamente tendrá dos variables de entrada. Por lo tanto, si hay un número impar de variables, la última variable será la primera entrada del último sistema. La Figura 1 muestra las jerarquías obtenidas para tres (a) y cuatro (b) variables. Dicho de una manera más formal, un SJPBRD codificado según la forma explicada con X variables de entrada necesitaría $X-1$ SBRDs simples.

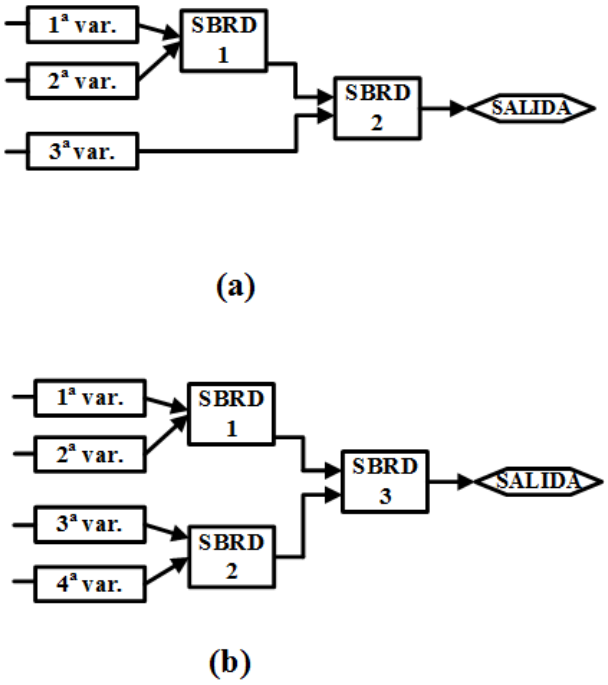

Fig. 1.SJBRD Paralelo estructurado para tres variables (a) y cuatro variables (b)

En el trabajo actual, se han utilizado SBRDs de tipo Takagi-Sugeno-Kang [15], usando trapecios para la codificación de las entradas y singletons (constantes) para las salidas. Para la inferencia se ha utilizado una T-norma mínima y agregación por centro de masas.

B. Algoritmos Genéticos

Los Algoritmos Genéticos son meta-heurísticas de búsqueda que imitan los procesos de selección natural de las especies. En su versión más clásica, un AG mantiene una población de soluciones candidatas, o individuos. Cuatro procesos son aplicados iterativamente en dicha población:

1. Operador de selección: Este proceso selecciona dos (o más) individuos para ser padres, dependiendo de su fitness.
2. Operador de cruce: Este operador crea nuevos individuos combinando los padres.
3. Operador de mutación: Este proceso modifica un individuo aplicando algún tipo de cambio pequeño en su representación.
4. Método de reemplazo: Este proceso depende de cómo se introducen nuevos individuos en la población.

AGs han sido utilizados en muchos casos para mejorar o *afinar* diferentes componentes de un SBRD [16], [17].

C. Entropía Cruzada

El método de EC fue propuesto por Rubinstein en 1997 [7]. Fue creado como un método adaptativo para eventos que ocurrieran con probabilidades muy

bajas, también llamados *rare-event propabilities*, y para optimización combinatoria. EC tiene tres fases principales:
1. Generación de un número $Ejemplos_{num}$ de ejemplos aleatorios siguiendo una distribución normal con una media y una desviación típica $\bar{x}$ y σ respectivamente.
2. Seleccionar un número $Ejemplos_{seleccionados}$ de mejores ejemplos a partir del conjunto creado en la fase anterior.
3. Actualizar los valores de $\bar{x}$ y σ de acuerdo con el fitness obtenido por los mejores ejemplos.

Conforme el algoritmo avanza, $\bar{x}$ se mantiene en los puntos con mejores resultados mientras que σ disminuye hasta que ambos se centran en el área de mejores soluciones encontradas en el dominio.

En el Algoritmo 1 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.

Algoritmo 1 Pseudocódigo del proceso seguido por la Entropía Cruzada

Entrada: $Ejemplos_{num}, Ejemplos_{actualizar}$, $Learn_{rate}$
Salida: Mejor individuo encontrado
1: $\overline{x} \leftarrow$ Inicializar Medias
2: $\sigma \leftarrow$ Inicializar Desviaciones
3: **while** Condición de parada no alcanzada
4: $Ejemplos \leftarrow$ Genera $Ejemplos_{num}$ 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: $\sigma \leftarrow (1 - Learn_{rate}) \cdot \sigma + Learn_{rate} \cdot$ Desv($Ejemplos_{seleccionados}$)
9: **fin while**

EC se puede aplicar en problemas de estimación u optimización [18], y puede ser usado en diferentes campos [19].

III. Algoritmo Genético con Entropía Cruzada: GACE

En este trabajo, se optimiza un SJPBRD con un algoritmo híbrido que combina tanto un AG como un método de EC, que recibe el nombre de GACE (*Genetic Algorithm and Cross Entropy*). Este sistema se usará para predecir la congestión a corto plazo en una vía. Por un lado, el uso de un SJPBRD ayuda con el problema de la dimensionalidad mientras que, por otro lado, GACE mejora la selección de variables para cada sistema y el cálculo de las etiquetas y reglas difusas de cada uno. La explicación del algoritmo se presenta en esta sección.

Este capítulo está estructurado como sigue: En la Sección III-A se expone la estructura del cromosoma y el cálculo de la función fitness. La Sección III-B contiene la explicación del algoritmo GACE y su funcionamiento. Por último, la Sección III-C contiene los operadores utilizados en la parte del Algoritmo Genético mientras que la Sección III-D contiene los operadores utilizados en la Entropía Cruzada.

A. Estructura del cromosoma y cálculo del fitness

El cromosoma codifica las tres partes de cada uno de los SBRD que componen un SJPBRD:
1. Jerarquía: se define como el subconjunto de variables seleccionadas para ser procesadas por el SJPBRD, así como el orden en el que estas deben ser incluidas en el sistema.
2. Funciones de pertenencia: contiene la localización de las MFs utilizados para codificar cada una de las variables de cada sistema SBRD en la jerarquía.
3. Base de reglas: codifica las posiciones de los *singletons* usados como consecuentes de la base de reglas de un sistema SBRD en la jerarquía.

A lo largo de este artículo, estas partes se nombrarán como $C_{jerarquia}$, $C_{etiquetas}$ 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 SJPBRD 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 $N_{variables}$, dado que el número máximo de $N_{modulos}$ es $N_{variables} - 1$. La Figura 1 ilustra 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 SBRD respectivamente. De manera formal, $C_{etiquetas}$ está formado por dos matrices de $N_{modulos} \cdot N_{variables}$ elementos. Por tanto, se cuenta con:

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

donde cada $MF_i(j,k)$ denota la k-ésima función de pertenencia MF usada para codificar la i-ésima entrada del j-ésimo SBRD en la jerarquía. Además, se usa una técnica denominada *lateral tuning* presentada en [20] 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 $[min, max]$.

Finalmente, se presenta C_{reglas} como una matriz de valores reales en el intervalo [0, 1]. Por cada sistema individual SBRD, su base de reglas RB tiene un tamaño de $N^2_{etiquetas}$. Hay que tener en cuenta que se trabaja con bases de reglas completas. Es decir, para cada entrada, y cada etiqueta de dicha entrada, se cuenta con una salida.

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

$$C_{reglas} = Reglas(i, j)$$
$$\forall_1 \in \{1 \dots N_{modulos}\},\ j \in \{1 \dots N^2_{etiquetas}\} \qquad (2)$$

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

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.

Para la función fitness, se ha utilizado el error medio absoluto (*mean absolute error*, MAE) [21] entre el nivel de congestión predicho por el sistema y el obtenido del conjunto de datos.

B. GACE: Algoritmo Genético y Entropía Cruzada

El algoritmo GACE está diseñado con la idea de aprovechar la habilidad de exploración de un AG y la habilidad de explotación de EC en un problema de optimización dado, con el objetivo de optimizar las entradas, etiquetas y reglas de un SJPBRD.

Con dicho objetivo, primero se realiza la inicialización de la población inicial que se utilizará en el SJPBRD. En cada iteración, la población de soluciones (POB_t) se divide en dos subpoblaciones, GA_{pob} y CE_{pob}, con GA_{tam} y CE_{tam} individuos respectivamente. Los individuos en GA_{pob} son elegidos a partir del método de selección escogido y se usan para aplicar los diferentes operadores de un AG. Por otra parte, la población CE_{pob} está formada por los CE_{tam} mejores individuos de la población y son usados en la parte de EC del algoritmo. Tanto GA_{tam} como CE_{tam} son elegidos por el usuario y su suma es igual al tamaño de la población (POB_{tam}).

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

- GA_{pob}: Los operadores de cruce y mutación se aplican a esta población para generar GA_{tam} nuevos individuos. Los operadores utilizados se especifican en la Sección III-C
- CE_{pob}: En este caso, el algoritmo EC se aplica a los individuos de esta población. Primero se actualizan $\bar{x}$ y σ. Tras esto, CE_{tam} individuos son generados aleatoriamente siguiendo una distribución normal usando $\bar{x}$ y σ. En la Sección III-D se presenta este proceso en detalle.

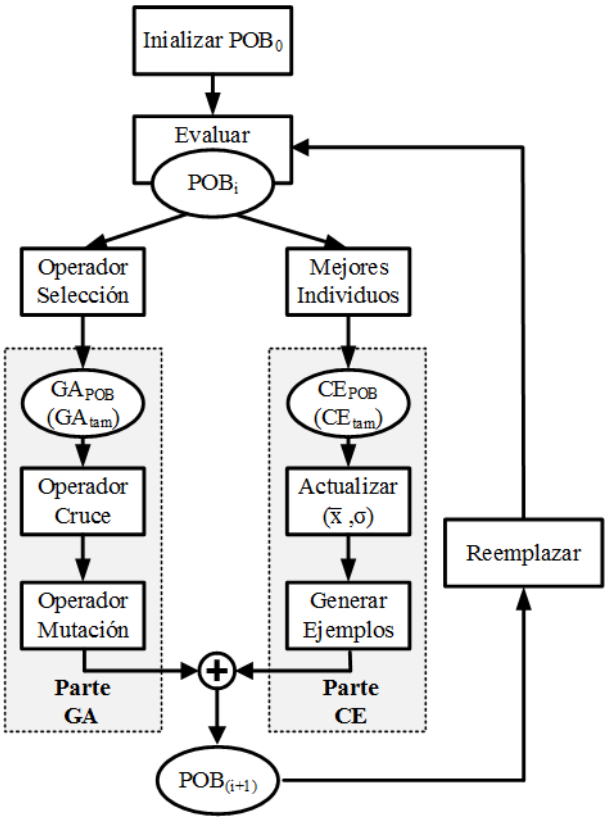

Fig. 2. Funcionamiento de GACE

Con los individuos de estas poblaciones se crea una población completamente nueva que contendrá GA_{tam} individuos de los operadores del AG y CE_{tam} individuos del método EC. Esta nueva población reemplaza completamente a la anterior.

Hay que tener en cuenta que para la optimización de un SJPBRD 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. El funcionamiento descrito en esta sección queda plasmado en la Figura 2.

C. Operadores del Algoritmo Genético

En esta sección se expone la explicación de los diferentes operadores usados en la parte del AG.

Mientras que como función de selección se utiliza el Torneo Binario [22], son necesarios dos operadores de cruce y otros dos de mutación debido a que las etiquetas y las reglas de los individuos son matrices que contienen valores reales, y la jerarquía es una permutación.

Para la permutación, se escoge una variante del Order Crossover [23], 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.

Para la parte real de los individuos (etiquetas y reglas), se ha escogido el método de cruce BLX-α [24]. Dado dos padres $A = (a_1 \dots a_m)$ y $B = (b_1 \dots b_m)$ por cada i, BLX-α crea dos hijos generando valores aleatorios en el intervalo presentado en la Ecuación 3 con $\alpha \in$ [0,1].

$$[min(a_i, b_i) - \alpha \cdot |a_i - b_i|,\ max(a_i, b_i) + \alpha \cdot |a_i - b_i|] \quad (3)$$

Respecto a los operadores de mutación, para la jerarquía, se aplica un operador de intercambio [25] mientras que para las etiquetas y las reglas, se aplica una mutación BGA [26]. Dado $A = (a_1 \dots a_m)$, el operador devuelve a'_i, calculado tal y como se presenta en la Ecuación 4

$$a'_i = a_i \pm \beta \cdot \sum_{k=0}^{15} (\alpha_k 2^{-k}) \quad (4)$$

donde β define el rango de mutación. El signo (+ o -) se escoge con probabilidad 0.5, $\alpha_k \in \{0, 0.33, 0.66, 1\}$ se genera aleatoriamente con $p(\alpha_k = \{0,\ 0.33,\ 0.66,\ 1\}) = \frac{1}{4}$.

D. Operadores de la Entropía Cruzada

Como se comentaba anteriormente, deben mantenerse dos individuos a lo largo del CE: uno que mantenga la media $\bar{x}$ y otro que guarde la desviación típica σ. Estos individuos tienen la misma estructura que un individuo normal 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 I. 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.

TABLA I
VALORES INICIALES POR CADA UNA DE LAS PARTES DE $\overline{x}$ Y σ.

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

Para la permutación ($C_{jerarquia}$) de los individuos, tanto la actualización de los valores media y desviación típica como la generación de los nuevos ejemplos sigue un proceso diferente. Los vectores que representan la jerarquía se convierten en vectores que guardan el orden de cada una de las variables. 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 3 y funciona de la siguiente manera: 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 desviación típica 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 desviación típica en (d) y se realiza el proceso inverso de transformación de vectores de orden a vectores de jerarquía (e).

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

IV. EXPERIMENTACIÓN

Esta sección contiene los resultados de los diferentes experimentos realizados para este artículo. La Sección IV-A contiene una explicación sobre los datos utilizados y los diferentes valores de congestión. En la Sección IV-B se expone la configuración de los diferentes experimentos. Por último, la Sección IV-C contiene los resultados obtenidos por dichos experimentos.

A. Datos utilizados

Los datos usados en este trabajo son proporcionados por el Caltrans Performance Measurement System (PeMS [1]). 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). Se han utilizado un total de 13 puntos donde los sensores están situados en la carretera principal y 8 sensores situados en rampas de entrada y salida. Las medidas de tráfico son recogidas por dichos sensores cada 5 minutos. La Figura 4 ilustra la carretera utilizada. 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.

Cada sensor en la carretera principal recoge la ocupación, fluidez y velocidad del tráfico. Por otro lado, los sensores situados en las rampas sólo recogen la fluidez. Se añade una variable llamada *congestión* al final de cada instancia. Se calcula usando los intervalos que se muestran en la Tabla II, la cual se obtiene de [27]. Por lo tanto, la congestión puede tomar cuatro valores: $\{Nula, Ligera, Moderada$ y $Severa\}$. Cada uno se representará por un valor real $\{0, 1/3, 2/3, 1\}$. 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: $densidad = fluidez/velocidad$.

[1] http://www.pems.dot.ca.gov

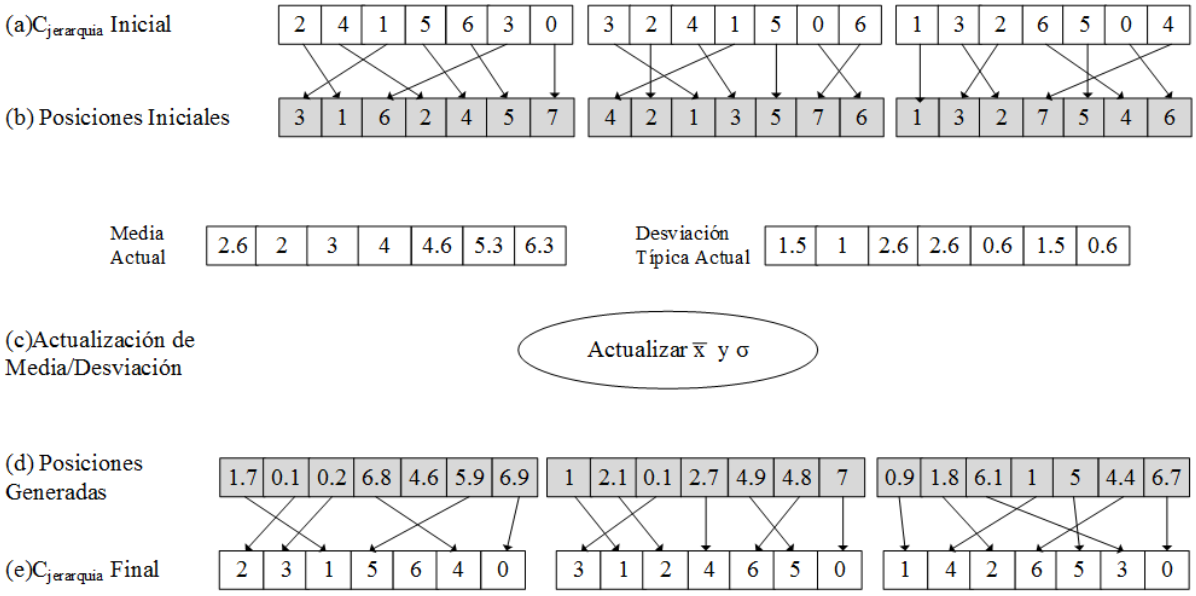

Fig. 3. Fases de creación de la parte $C_{jerarquia}$ en un nuevo individuo

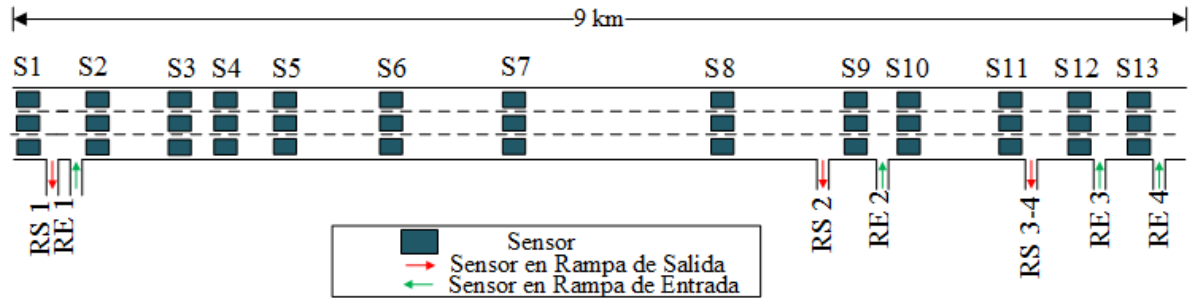

Fig. 4. Intervalo de 9 km de la carretera I5 con los sensores utilizados en este trabajo

TABLA II
VALORES DE CONGESTIÓN Y SU CÁLCULO.

Congestión	Densidad(ve/km/ln)	Velocidad(km/h)
Ligera	[29–37]	[48–80]
Moderada	[37–50]	[24–64]
Severa	> 50	< 40
Nula	Resto de casos	

Con los datos de los que se ha hablado en la sección anterior, se han creado dos 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 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.

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 $\{5, 15, 30\}$ minutos. Por lo tanto, se utilizarán un total de 6 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.

B. Configuración de la experimentación

Combinaciones diferentes del número de individuos en GA_{pob} y CE_{pob} son escogidas para realizar los experimentos y comparar sus resultados. Además, las configuraciones utilizadas en el GACE se comparan con un AG puro ($CE_{tam} = 0$) y con una EC puro ($GA_{tam} = 0$) para comprobar los beneficios de la hibridación realizada en comparación con 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 en 500 el número de generaciones y el tamaño total de la población es igual a 50. Las ejecuciones se nombran por el tamaño de sus poblaciones, es decir, $GA_{tam} - CE_{tam}$. Por ejemplo, 50-0 se refiere a un AG con los operadores explicados en esta memoria, y 0-50 corresponde a una ejecución de una EC. El tamaño de la población del AG 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 MF usado en cada uno de los sistemas SBRD 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), siendo este 48.

Para la parte de la experimentación correspondiente al AG, 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 EC, el $Learn_{rate}$ se ha establecido a 0.7 para actualizar tanto $\bar{x}$ como σ.

C. Resultados

Esta sección contiene los resultados de los diferentes experimentos realizados con el algoritmo presentado de forma que podamos comparar el rendimiento del mismo y encontrar su configuración óptima, a la vez que se compara con un AG y una EC.

La Tabla III presenta el porcentaje medio de error en los conjuntos de test. Los resultados resaltados indican los dos mejores valores para cada dataset.

Analizando dicha tabla, $GACE_{40-10}$ obtiene uno de los dos mejores resultados en 4 de 6 casos mientras que $GACE_{45-5}$ lo hace en 5 de 6. Por otra parte, tanto $GACE_{10-40}$ como CE no se encuentran nunca entre los mejores resultados. Comparando AG con EC, AG obtiene mejor resultado en todos los casos, mientras que EC obtiene el peor resultado en 2 de los 6 casos de estudio. En el resto de casos, $GACE_{15-35}$ y $GACE_{10-40}$ obtienen los peores resultados. Se puede decir, por tanto, que en la mayoría de los casos, un GACE con $GA_{tam} \in [35, 45]$ obtiene la mejor precisión frente a otros algoritmos como $GACE_{25-25}$, AG o EC.

TABLA III
ERROR MAE DE LOS CONJUNTOS DE TEST POR TÉCNICA Y DATASET

	AG	45-5	40-10	35-15	25-25	15-35	10-40	EC
DP_5	.014	**.009**	**.009**	.01	.012	.017	.017	.04
DP_{15}	.027	**.013**	**.013**	**.013**	.015	.014	.019	.028
DP_{30}	.018	.017	**.016**	**.016**	.018	.029	.021	.027
DS_5	**.174**	**.149**	.184	.195	.175	.206	.295	.2
DS_{15}	.176	**.13**	.182	**.169**	.198	.194	.231	.219
DS_{30}	.142	**.123**	**.14**	.169	.185	.216	.208	.198

Por otro lado, comparamos los mejores resultados obtenidos en los datasets con GA_{tam} superior a CE_{tam} con tres métodos de la literatura en la Tabla IV. En concreto:

- C4.5: Algoritmo que genera árboles de decisión y es una extensión del algoritmo ID3.
- One Rule: Clasificador que usa el error mínimo para predecir, discretizando atributos numéricos.
- LogitBoost: Clasificador que usa un esquema de regresión, concretamente el aprendizaje base y puede utilizarse con problemas multiclase.

Para todos estos algoritmos se utiliza sus parámetros por defecto establecidos por WEKA [2]. En dicha tabla se muestra como la técnica propuesta se encuentra entre las dos técnicas con menor error en 5 de los 6 casos de estudio, siendo superada de forma destacable por C4.5 en tres de los casos, concrétamente en los estudios realizados sobre el dataset Sector.

Las dos últimas filas de la tabla muestran la posición promedio que ocupa cada técnica dependiendo del dataset utilizado, siguiendo la primera parte del Test de Friedman [28]. Por ejemplo, en el caso de $GACE_{40-10}$ con cualquiera de los dataset DS, es la mejor situada seguida por $GACE_{45-5}$. En el caso de los datasets DP, la técnica mejor posicionada es $C4{,}5$ seguida por $GACE_{45-5}$.

TABLA IV
COMPARATIVA DE MENOR ERROR OBTENIDO POR CADA UNO DE LOS ALGORITMOS

	45-5	40-10	35-15	C4.5	OneR	LogitBoost
DP_5	**.008**	**.008**	.009	.011	.010	.010
DP_{15}	**.012**	**.012**	.013	.013	.015	**.011**
DP_{30}	.015	**.014**	.015	.016	**.010**	.016
DS_5	.072	.108	.096	**.014**	.104	**.061**
DS_{15}	**.085**	.112	.134	**.064**	.135	.086
DS_{30}	.094	.094	**.088**	**.076**	.138	.101
DP_*	2.50	**2.00**	3.67	5.33	3.83	3.67
DS_*	2.83	4.50	3.67	**1.00**	5.67	3.33

V. CONCLUSIONES

En este trabajo se ha presentado una combinación de Algoritmo Genético y un método de Entropía Cruzada para la optimización de una Jerarquía en Paralelo de Sistemas Basados en Reglas Difusas. El objetivo de esta combinación es obtener una sinergia entre la exploración y la explotación de manera que se pudiera mejorar los parámetros de los sistemas. El método divide la población en dos subpoblaciones. En una de ellas, se aplica un Algoritmo Genético mientras que en la otra se ejecuta un método de Entropía Cruzada. Después de esto, se unen ambas poblaciones y reemplazan la población actual. El objetivo final es predecir congestión en un punto o tramo de la carretera I5 en California en intervalos de tiempo de 5, 15 y 30 minutos. Los datos han sido obtenidos a partir de una fuente gubernamental y usados para probar la eficiencia de los sistemas. Con ellos se han creado seis datasets, conteniendo

[2] http://www.cs.waikato.ac.nz/ml/weka

datos de congestión tanto de un punto como de un segmento de carretera.

Se han probado diferentes configuraciones de tamaños de población a usar por cada parte de la hibridación. Las configuraciones del GACE han sido comparadas con los dos algoritmos por separado para comprobar los beneficios de usar la hibridación propuesta, en comparación con los métodos aplicados individualmente. El resultado es un buen rendimiento por parte del algoritmo propuesto, dado que los sistemas pueden predecir congestión a corto plazo en un punto con un error muy pequeño. GACE obtiene mejores resultados con un mayor peso en la parte del AG. Esta afirmación es respaldada por las ejecuciones de $GACE_{45-5}$ y $GACE_{40-10}$, que obtienen en 5 y 4 de los 6 casos de estudio respectivamente un mayor rendimiento que el resto. Por otro lado, una Entropía Cruzada pura obtiene el peor resultado en 4 de los 6 casos. Además, con dichas configuraciones, el algoritmo se situa entre las dos primeras técnicas en 5 de los 6 casos de estudio cuando se compara con técnicas de la literatura.

Como líneas de trabajo futuras, se piensa en realizar una mejora de la optimización de la jerarquía de sistemas SJPBRD aplicando otros métodos, así como un modelo que detecte automáticamente las mejores combinaciones de tamaños de población para ambas partes, o mejore los resultados en los casos de congestión moderada y severa. Además, se tendrá en cuenta para trabajos futuros la comparación de esta técnica con técnicas como Redes Neuronales, Árboles de Decisión o incluso métodos de *Clustering*.

Referencias

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] R.Y. Rubinstein, "Optimization of computer simulation models with rare events," *European Journal of Operational Research*, vol. 99, no. 1, pp. 89–112, 1997.

[8] Lotfi A. Zadeh, "Soft computing and fuzzy logic," *IEEE Software*, vol. 11, no. 6, pp. 48–56, 1994.

[9] 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.

[10] Lotfi A Zadeh, "Fuzzy sets," *Information and control*, vol. 8, no. 3, pp. 338–353, 1965.

[11] 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*, Aug 2009, pp. 1615–1620.

[12] 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.

[13] Alberto Fernández, María José del Jesus, and Francisco Herrera, "Hierarchical fuzzy rule based classification systems with genetic rule selection for imbalanced datasets," *International Journal of Approximate Reasoning*, vol. 50, no. 3, pp. 561–577, 2009.

[14] 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.

[15] Pei-Chann Chang and Chen-Hao Liu, "A tsk type fuzzy rule based system for stock price prediction," *Expert Systems with Applications*, vol. 34, no. 1, pp. 135–144, 2008.

[16] 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.

[17] 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.

[18] Reuven Rubinstein, "The cross-entropy method for combinatorial and continuous optimization," *Methodology and computing in applied probability*, vol. 1, no. 2, pp. 127–190, 1999.

[19] 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 Technology and Management*, vol. 20, no. 1-4, pp. 316–330, 2010.

[20] 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.

[21] Cort J Willmott and Kenji Matsuura, "Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance," *Climate Research*, vol. 30, no. 1, pp. 79, 2005.

[22] David E Goldberg and Kalyanmoy Deb, "A comparative analysis of selection schemes used in genetic algorithms," *Urbana*, vol. 51, pp. 61801–2996, 1991.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] Cox Skycomp, Inc. in association with Whytney Bailey and LLC Magnani, "Major highway performance ratings and bottleneck inventory, state of maryland - spring 2008," 2009.

[28] Janez Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.