Un algoritmo memético para selección de prototipos: Una propuesta eficiente para problemas de tamaño medio

Salvador García, José Ramón Cano, Francisco Herrera

Resumen— **Dentro del marco del aprendizaje basado en instancias existen clasificadores basados en similaridad entre muestras, como es el caso de la regla del vecino más cercano. La Selección de Prototipos es una fase optativa previa al clasificador que consiste en reducir el conjunto de instancias utilizadas como referencia para clasificar con el fin de eliminar aquellas muestras ruidosas o redundantes y acelerar el proceso de clasificación.**

Los Algoritmos Evolutivos son técnicas de optimización que se han usado recientemente en la selección de prototipos con resultados excelentes. Dentro de la misma familia de algoritmos, existen los llamados Algoritmos Meméticos, que combinan algoritmos basados en poblaciones con búsquedas locales. En este trabajo, proponemos un modelo de algoritmo memético que incorpora una búsqueda local especialmente diseñada para el problema de la selección de prototipos. Con el fin de comprobar su eficacia, hemos llevado a cabo un estudio empírico que incluye una comparación entre nuestra propuesta y otros métodos evolutivos para seleccionar prototipos estudiados en la literatura, ejecutados sobre conjuntos de datos de un tamaño considerable.

Los resultados han sido contrastados con procedimientos estadísticos no paramétricos que nos muestran como nuestra propuesta mejora el comportamiento de los modelos anteriores, considerando como objetivos la eficacia en acierto y la tasa de reducción con respecto al conjunto original.

Palabras clave— **Selección de prototipos, algoritmos meméticos, reducción de datos, minería de datos.**

I. Introducción

La regla del vecino más cercano (NN: Nearest Neighbour) es un método de clasificación supervisado clásico utilizado en reconocimiento de patrones [20], [22]. El clasificador del vecino más cercano intenta predecir la clase de un nuevo prototipo calculando la distancia entre él y todos los demás prototipos almacenados (en nuestro caso, hacemos uso de la distancia euclídea), y considera como respuesta a la clase del más cercano (en el caso del clasificador 1-NN) [1].

Estudios recientes muestran que el clasificador k-NN puede ser muy competitivo con respecto al conjunto de métodos de clasificación que se encuentran en el estado-del-arte. Además, la regla k-NN podría ser mejorada desarrollando funciones de similaridad más potentes [24], las cuales definen la vecindad de un patrón, o aprendiendo pesos para minimizar el error de clasificación [21].

Otra forma de mejorar un clasificador k-NN consiste en reducir el número de ejemplos para entrenamiento mientras se mantiene la calidad del clasificador. Este procedimiento puede ser efectuado proporcionando al algoritmo de aprendizaje cierto control sobre las entradas y seleccionando los prototipos más apropiados para clasificar nuevos ejemplos [16]. Cuando el proceso de reducción del número de patrones debe ser independiente del clasificador, se pueden emplear dos alternativas: generación o condensación de prototipos [3] y Selección de Prototipos (SP).

Como antes hemos introducido, la SP es un problema de aprendizaje supervisado clásico en donde el objetivo consiste en, usando un conjunto de datos como entrada, encontrar aquellos prototipos que mejoran el acierto del clasificador basado en vecinos cercanos [20], [22], [25]. Más formalmente, asumimos la existencia de un conjunto de entrenamiento T compuesto por parejas $(x_i, y_i), i = 1, ..., n$, en donde x_i define un vector de entrada de atributos e y_i define la etiqueta de clase correspondiente. T contiene n ejemplos, los cuales tienen cada uno m atributos de entrada y deberían pertenecer a una de las c clases. El resultado de la ejecución de un algoritmo de SP consistirá en un subconjunto de ejemplos seleccionados $S \subseteq T$ cuyo objetivo es mejorar el comportamiento de la regla NN.

En la literatura especializada, existen numerosas propuestas de algoritmos de SP [25], [10], entre las cuales existe un grupo que pertenece a la familia de los Algoritmos Evolutivos (AEs), Selección de Prototipos Evolutiva (SPE), en la cual se citan propuestas recientes que nos ofrecen resultados prometedores [4]. Generalmente, los AEs clásicos sufren una convergencia excesivamente lenta debido a su falta de habilidad de explotación local de las soluciones. Esto es una causa que hace que los AEs estén limitados a ser utilizados bajo condiciones de excesivo tamaño de los datos debido a los altos tiempos de cálculo esperados. Una manera de subsanar esta falta de explotación local consiste en la combinación de AEs con Búsquedas Locales (BL), lo que fue denominado en [19] como *Algoritmos Meméticos* (AMs). Formalmente, un AM se define como un AE que incluye una o más fases de búsqueda local dentro de su ciclo evolutivo [13]. La elección del nombre está ins-

S. García y F. Herrera pertenecen al Dept. de Ciencias de la Computación e Inteligencia Artificial. ETS Ingeniería Informática. Univ. de Granada. 18071 Granada. E-mail: {salvagl,herrera}@decsai.ugr.es. J.R. Cano pertenece al Dept. de Ciencias de la Computación. Univ. de Jaén. 23700 Linares, Jaén. E-mail: jrcano@ujaen.es

pirada en el concepto de *meme*, que representa a la unidad de evolución cultural que puede mostrar refinamiento local [6]. Los AMs se han mostrado más eficientes (p.e. necesitar de menos evaluaciones para encontrar el óptimo) y más efectivos (encontrar soluciones de mayor calidad) que los AEs tradicionales en algunos dominios de problemas.

En este trabajo, nos proponemos diseñar un AM que utilizará un meme específico para el problema de SP, aprovechándonos de que el problema de SP es de naturaleza divisible y de la simplicidad de hibridación de un procedimiento de optimización local dentro de un AE. Como objetivo de este trabajo, junto a presentar la nueva propuesta, nos planteamos un estudio comparativo con otros métodos evolutivos de SP que podemos encontrar en la literatura especializada.

Este trabajo se organiza de la siguiente manera. La Sección II explica los características básicas de la SPE. La explicación del algoritmo memético propuesto y del procedimiento meme se encuentra en la Sección III. En la Sección IV se detallan los aspectos del estudio empírico llevado a cabo y se presenta el análisis experimental. La Sección V enumera las conclusiones finales.

II. Selección de Prototipos Evolutiva

Los AEs [8] son métodos de búsqueda estocásticos que imitan la metáfora de la evolución natural biológica. Todos los AEs cuentan con el concepto de *población* de individuos (que representan puntos de búsqueda en el espacio potencial de soluciones a un problema dado), la cual puede someterse a operadores probabilísticos como la *mutación*, *selección* y la *recombinación*. El *fitness* de un individuo refleja su valor de la función objetivo con respecto a una función objetivo particular a ser optimizada. El operador de mutación introduce innovación dentro de la población, el operador de recombinación efectúa un intercambio de información entre individuos de una población y el operador de selección impone una obligación dirigida dentro del proceso de evolución prefiriendo los mejores individuos para que sobrevivan y se reproduzcan.

El problema de la SP puede ser considerado como un problema de búsqueda en el cual los AEs pueden ser aplicados. Diferentes propuestas de AEs se han estudiado en la literatura, las describiremos en el resto de la sección.

Un diseño de Algoritmo Genético (AG) para obtener un clasificador de vecinos cercanos óptimo es propuesto en [12]. Se trata un Algoritmo Genético Inteligente (IGA) que usa como representación de soluciones un esquema de codificación binario y está basado en el diseño de experimentos Ortogonal [15] usado para SP y Selección de Características. IGA es un AG Generacional (GGA) que incorpora un operador de Cruce Inteligente (IC). EL IC construye un Array Ortogonal (OA) (ver [12]) a partir de dos padres y busca dentro del OA los dos mejores individuos de acuerdo a la función de fitness. Se toma unas $2^{\lceil \log_2(\gamma+1) \rceil}$ evaluaciones para efectuar una operación de IC, en donde γ es el número de bits que difieren entre los dos padres. Nótese que solo una aplicación de IC en cromosomas de gran tamaño (resultantes de conjuntos de tamaño grande) podría consumir un alto número de evaluaciones. La función fitness utilizada en IGA mide la calidad de los cromosomas según el número de prototipos que pueden clasificar correctamente utilizando el subconjunto de instancias que representan; por tanto, el propósito de IGA es maximizar dicha función fitness.

El término SPE empezó a ser adoptado en [4], en donde se analiza el comportamiento de diferentes AEs, AG estacionario (SSGA), GGA, modelos CHC [9] y PBIL [2]. A continuación, detallamos los dos aspectos comunes de los cuatro modelos anteriores: la especificación de la representación de las soluciones y la definición de la función de fitness.

- *Representación*: Asumimos un conjunto de datos denotado por T con n instancias. El espacio de búsqueda asociado está constituido por todos los subconjuntos de T. Esto se consigue usando una representación binaria. Un cromosoma está compuesto de n genes (uno por cada instancia de T) con dos posibles estados: 0 y 1. Si el gen es 1, la instancia asociada se incluye en el subconjunto de T representado por el cromosoma. Si es 0, la instancia asociada no pertenece al subconjunto.
- *Función de Fitness*: Sea S un subconjunto de instancias de T para evaluar y estar codificado por un cromosoma. Definimos una función de fitness que combina dos valores: la tasa de clasificación ($tasa\_clas$) asociado con S y el porcentaje de reducción ($porc\_red$) de instancias de S con respecto a T.

$$Fitness(S) = \alpha \cdot tasa\_clas + (1-\alpha) \cdot porc\_red. \quad (1)$$

El clasificador 1-NN se utiliza para medir la tasa de clasificación conseguida con S. Esta tasa indica el porcentaje de objetos de T correctamente clasificados usando S solamente para encontrar el vecino más cercano. Para cada objeto y en S, el vecino más cercano se busca entre todos en el conjunto $S \setminus \{y\}$. Por otro lado, $porc\_red$ se define como

$$porc\_red = 100 \cdot \frac{|T| - |S|}{|T|}. \quad (2)$$

El objetivo de los AEs consiste en maximizar la función fitness definida; esto es, maximizar la tasa de clasificación y minimizar el número de instancias obtenidas.

III. Propuesta de Algoritmo Memético para Seleccionar Prototipos

En esta sección, presentamos nuestro modelo de AM para SPE. Se trata de un AM Estacionario (SS-

MA) que utiliza un procedimiento de Búsqueda Local (BL) o meme desarrollado específicamente para este propósito.

En la sucesivas secciones, introduciremos los fundamentos de los SSMAs, explicaremos con detalle el algoritmo propuesto y aclararemos la aplicación del procedimiento meme dentro del algoritmo.

A. Algoritmos Meméticos Estacionarios

En los SSGAs, normalmente uno o dos hijos se producen en cada generación. Los padres se seleccionan para producir hijos y después se toma una decisión acerca de los individuos de la población que se van a borrar para hacer hueco a los nuevos hijos. Los pasos básicos del algoritmo genético estacionario están descritos en la Figura 1.

En el paso 4, se puede elegir la *estrategia de reemplazamiento* (p.e., reemplazamiento del peor, del más antiguo, o un individuo elegido al azar). En el paso 5, se puede elegir la *condición de reemplazamiento* (p.e., reemplazamiento si el nuevo individuo es mejor, o reemplazamiento incondicional). Una combinación ampliamente utilizada es sustituir el peor individuo sólo si el nuevo es mejor. Llamaremos a esta estrategia la *estrategia estándar de reemplazamiento*. En [11], se sugiere que el borrado de los peores individuos induce una alta presión selectiva, incluso cuando los padres se seleccionen de forma aleatoria.

Aunque los SSGAs son menos comunes que los GGAs, diferentes autores [14], [17] recomiendan el uso de SSGAs para el diseño de AMs porque permiten que los resultados de la BL se mantengan en la población de generación en generación.

Los SSMAs integran la búsqueda global y local de forma más fina que los AMs Generacionales. Este entrelazamiento entre las fases de búsqueda global y local permiten que entre ambas exista influencia mutua; p.e., el SSGA elige buenos puntos de comienzo y la BL proporciona una representación más precisa de esa región del dominio. Por el contrario, los AMs Generacionales actúan alternando las fases de búsqueda global y local. Primero, el GGA produce una nueva población, y después el procedimiento meme se ejecuta. El estado específico del meme no se mantiene generalmente de una generación a otra, aunque los resultados del meme influyen en la selección del individuo.

1. Seleccionar dos padres de la población.
2. Crear un hijo usando cruce y mutación.
3. Evaluar el hijo de acuerdo a la función de fitness.
4. Seleccionar un individuo de la población, el cual debe ser sustituido por el hijo.
5. Decidir si este individuo será sustituido.

Fig. 1. Pseudocódigo para el modelo SSGA

B. Modelo de Algoritmo Memético Estacionario para el Problema de la Selección de Prototipos

Las principales características de nuestro SSMA son:

- *Inicialización de la Población*: El primer paso que el algoritmo efectúa consiste en la inicialización de la población, que se hace generando una población de cromosomas de forma aleatoria.
- *Representación*: Utilizamos el mismo modelo de representación que se ha utilizado en la SPE, se puede ver en la Sección II.
- *Función de Fitness*: Supongamos S como un subconjunto de instancias de T para evaluar y que está codificado en un cromosoma. Definimos una función de fitness considerando el número de instancias correctamente clasificadas usando el clasificador 1-NN. Para cada objeto y en S, el vecino más cercano es buscado entre todas en el conjunto $S \setminus \{y\}$.

$$Fitness(S) = N\_Instancias\_Bien\_Clasificadas$$

El objetivo del SSMA consiste en maximizar la función de fitness definida.

- *Mecanismo de Selección de Padres*: Para seleccionar dos padres con el objetivo de aplicar los operadores evolutivos, se emplea la selección por torneo binario.
- *Operadores Genéticos*: Usamos un operador de cruce que sustituye aleatoriamente el 20 % de los bits del primer padre con el segundo y viceversa. El operador de mutación supone una probabilidad para que un bit arbitrario de la secuencia genética pueda ser cambiado a su otro estado.
- *Estrategia de Reemplazamiento*: Nuestro SSMA usará un reemplazamiento de los peores individuos de la población en todos los casos. Esto significa que, aunque los peores individuos de la población sean mejores que los nuevos recién generados, siempre van a ser sustituidos por los hijos.
- *Condición de Parada*: El SSMA continúa haciendo iteraciones del ciclo hasta alcanzar un número determinado de evaluaciones.

La Figura 2 muestra el pseudocódigo del SSMA.

1. Inicializar población.
2. Mientras (*no se cumpla condición de parada*) hacer
   3. Usar Torneo Binario para seleccionar dos padres
   4. Aplicar el operador de cruce dos veces para crear los hijos (Off_1, Off_2)
   5. Aplicar mutación a Off_1 y Off_2
   6. Evaluar Off_1 y Off_2
   7. Efectuar la optimización local en Off_1 y Off_2
   8. Sustituir los dos peores individuos por Off_1 r Off_2.
9. Devolver el mejor cromosoma

Fig. 2. Pseudocódigo para el SSMA propuesto

C. Procedimiento Meme

En esta sección explicamos este procedimiento de optimización de acuerdo a la siguiente estructura:

primero, presentamos el mecanismo de evaluación con evaluaciones totales y parciales; segundo, presentamos el pseudocódigo describiendo el procedimiento completo; en tercer lugar, explicaremos las dos estrategias usadas asociadas a la mejora de fitness, fase de mejora en acierto y fase para evitar la convergencia prematura con pérdida del objetivo local; finalmente, dispondremos de un ejemplo que nos ayude a entender el procedimiento.

C.1 Mecanismos de Evaluación

Durante la operación del SSMA, tendrán lugar un número determinado de evaluaciones para determinar la calidad de los cromosomas. Podemos distinguir entre *Evaluación Total* y *Evaluación Parcial*.

- *Evaluación Total*: Consiste en una evaluación estándar de la calidad del cromosoma en la SPE, que conlleva calcular el vecino más cercano de cada instancia, que debe pertenecer al subconjunto seleccionado y llevar la cuenta de las instancias clasificadas correctamente. Las evaluaciones totales siempre tienen lugar fuera del procedimiento de optimización, esto es, dentro del ciclo evolutivo.
- *Evaluación Parcial*: Puede ser realizada cuando se actúa sobre una solución vecina de una ya evaluada y difiere solamente en una posición de bit, el cual ha cambiado del valor 1 al 0. Si una evaluación total cuenta como una evaluación en términos de llevar la cuenta del número de evaluaciones totales para la condición de parada, una evaluación parcial cuenta como:

$$EP = \frac{N_{nu}}{|T|} \tag{3}$$

en donde N_{nu} es el número de vecinos actualizados cuando una instancia determinada se borra por el procedimiento meme y $|T|$ es el tamaño del conjunto original de instancias (es también el tamaño del cromosoma). Las evaluaciones parciales tienen siempre lugar en el procedimiento de optimización local.

El SSMA calcula evaluaciones totales, cuando consideramos una evaluación parcial añadimos a la variable contador de evaluaciones el respectivo valor parcial EP (expresión 3). Por lo tanto, un número variable de evaluaciones parciales (depende de los valores dados por EP) se considerará como una evaluación total.

C.2 Descripción del Procedimiento de Optimización

El procedimiento de optimización meme usado en este método es un proceso iterativo que se enfoca a mejorar individuos de una población reduciendo el número de prototipos seleccionados e incrementando el acierto de clasificación.

El pseudocódigo descrito en la Figura 3 corresponde al procedimiento meme. A continuación, lo describimos.

Para alcanzar el doble objetivo (mejorar el número de patrones clasificados y reducir el tamaño del

1. Sea $S = \{s_1, s_2, ..., s_n\}$ un cromosoma a optimizar
2. Sea $R = \emptyset$
3. Sea $U = \{u_1, u_2, ..., u_n\}$ una lista de vecinos asociados en donde $u_i = Vecino\_Cercano_S(i)/i = 1, 2, ..., n$
4. Mientras ($\exists s_i \in S/s_i = 1$ y $i \notin R$) hacer
   5. Elegir aleatoriamente j de S en donde $s_j = 1$ y $j \notin R$
   6. $ganancia = 0$
   7. $s_j = 0$
   8. Copiar U a U'
   9. Para cada $u_i \in U/u_i = j$ hacer
      10. $u_i = Vecino\_Cercano_S(i)$
      11. si ($cls(i) = cls(u'_i)$ y $cls(i) \neq cls(u_i)$)
         12. $ganancia = ganancia - 1$
      13. si ($cls(i) \neq cls(u'_i)$ y $cls(i) = cls(u_i)$)
         14. $ganancia = ganancia + 1$
   15. si ($ganancia >= umbral$)
      16. $fitness_S = fitness_S + ganancia$
      17. $R = \emptyset$
   18. en otro caso
      19. Recuperar U de U'
      20. $s_j = 1$
      21. $R = R \bigcup j$
22. Devolver S

Fig. 3. Pseudocódigo de la optimización meme

subconjunto seleccionado) se consideran soluciones pertenecientes a la vecindad de la solución actual aquellas que tienen $m - 1$ instancias seleccionadas, siendo m igual al número de instancias seleccionadas actualmente (posiciones con valor 1 en el cromosoma). En otras palabras, un vecino es obtenido cambiando un 1 por un 0 en un gen. De esta forma, el número de ejemplos representado en el cromosoma después de efectuar una optimización siempre será menor o igual al número inicial de ejemplos seleccionados en el cromosoma original.

A continuación, explicamos los conceptos necesarios para entender el procedimiento. Dos funciones son muy útiles en el mismo:

- $Vecino\_Cercano_S(\cdot)$: Devuelve el vecino más cercano de una instancia considerando solamente las instancias seleccionadas por el cromosoma S. Esta función requiere como entrada un número entero que será el identificador de una instancia dentro del conjunto de entrenamiento T, compuesto por n instancias. La salida será también un entero identificador de la instancia más cercana a la entrada perteneciente al subconjunto S (o a las instancias seleccionadas en el cromosoma).
- $cls(\cdot)$: Devuelve la clase a la que pertenece la instancia

Además de esto, se define una estructura llamada U compuesta por una lista de identificadores de instancias. U tiene espacio para almacenar n identificadores y relaciona la instancia identificada por el número almacenado en la celda i-ésima como el vecino más cercano a la instancia identificada por el número i.

Una evaluación parcial puede aprovecharse de U y de la naturaleza divisible del problema de la SP cuando las instancias se van eliminando.

La variable llamada *ganancia* mantiene una con-

tabilización de las aportaciones de la búsqueda local llevada a cabo cuando se realiza un movimiento en el procedimiento meme. Su valor puede ser tanto negativo como positivo, dependiendo de si el vecino más cercano de una instancia cambia la etiqueta de clase con respecto al anterior vecino más cercano. Una aportación negativa, resta 1 al objetivo local, se produce cuando el nuevo vecino clasifica mal una instancia que antes se clasificaba bien. Una aportación positiva, suma 1 al objetivo local, se produce cuando el nuevo vecino clasifica bien una instancia mal clasificada anteriormente. Una aportación nula se produce cuando la instancia mantiene el mismo estado de clasificación, sigue siendo mal clasificada o sigue siendo bien clasificada. Este proceso se lleva a cabo sobre todas las instancias en las cuales su vecino más cercano ha cambiado, haciendo uso de los identificadores de los vecinos cercanos almacenados en la estructura U, apropiadamente actualizada tras un movimiento del procedimiento meme.

C.3 Etapas de la Búsqueda Local

Dos etapas pueden ser diferenciadas dentro del proceso de optimización. Cada una tiene un objetivo diferente y su aplicación depende del progreso del proceso de búsqueda global: la primera de ellas es una mejora exclusiva del fitness y la segunda etapa es una estrategia para tratar con el problema de la convergencia prematura.

- *Etapa de Mejora del Acierto*: Comienza desde una asignación inicial (un hijo generado recientemente) e intenta mejorar iterativamente la asignación actual mediante cambios locales. Si en la vecindad de la asignación actual, se encuentra una asignación mejor, sustituye la actual y se continúa desde esta nueva asignación. La selección de un vecino se hace aleatoriamente sin repetición entre todas las soluciones que pertenecen a la vecindad. Para considerar una asignación mejor que la actual, el valor de fitness debe ser mayor o igual que ésta, teniendo en cuenta que, en el caso de igualdad, el número de instancias seleccionadas será menor que en la anterior asignación.
- *Etapa para Evitar la Convergencia Prematura*: Cuando el proceso de búsqueda avanza, tiene lugar una tendencia de la población a la convergencia prematura hacia una cierta área del espacio de búsqueda. Una optimización local acentúa este comportamiento cuando se consideran soluciones con mejor acierto en clasificación. Para prevenirnos de esta conducta, el meme propuesto aceptará peores soluciones en la vecindad si se cumplen dos condiciones: la diferencia del fitness entre la solución actual y la nueva no será mayor que una unidad y un número fijado de evaluaciones ya ejecutadas ha sido sobrepasado (consideramos sobrepasar la mitad del número total de ellas).

El parámetro *umbral* se utiliza para determinar el modo de operación del algoritmo con respecto a la etapa actual en la que se encuentre. Cuando *umbral* tiene un valor de 0, entonces la etapa en curso es *Etapa de Mejora del Acierto* porque un recién generado cromosoma vía optimización meme se acepta cuando su aportación al fitness no es negativa ($ganancia >= 0$). Nótese que una $ganancia = 0$ implica que el nuevo cromosoma es tan bueno como el original considerando el objetivo de acierto en clasificación, pero tendrá una instancia menos (lo que implica mejorar el objetivo de reducción). Por otro lado, si *umbral* tiene un valor de -1, entonces la etapa en curso es *Etapa para Evitar la Convergencia Prematura* porque un nuevo cromosoma se acepta cuando su aportación al fitness es negativa, pero solo una unidad ($ganancia >= -1$).

C.4 Ejemplo del Meme

Mostramos un ejemplo en la Figura 4, en donde se considera un cromosoma de 13 instancias. El procedimiento meme borra la instancia número 3. Una vez borrada, la instancia 3 no puede aparecer dentro de la estructura U como vecino cercano de otra instancia. U debe ser actualizada en este momento obteniendo los nuevos vecinos cercanos para las instancias que tenían la instancia número 3 como vecino cercano. Entonces, se calcula un objetivo relativo con respecto al cromosoma original (la ganancia de las instancias 1, 5, 11 y 13). El resultado obtenido es un nuevo cromosoma con un número de patrones correctamente clasificados mayor que el cromosoma original (8 en vez de 7) con un consumo de evaluaciones de 4/13, el cual será sumado a la cuenta total del número de evaluaciones.

IV. Estudio Experimental

En esta sección, analizaremos el comportamiento de la propuesta de SSMA para SPE usando 8 conjuntos de datos escogidos del Repositorio de Bases de Datos para Aprendizaje Automático UCI [18]. Este análisis se llevará a cabo mediante una comparación de los resultados obtenidos por nuestro algoritmo y el resto de algoritmos de SPE explicados en la sección II.

Las bases de datos utilizadas en este estudio son consideradas de tamaño mediano o grande para la SP. Cuando consideramos conjuntos de tamaño grande, se usa el proceso de estratificación [5] para obtener estratos de tamaño mediano. La Tabla I resume las principales características de las bases de datos utilizadas. Independientemente del tamaño de los conjuntos, todos los algoritmos serán ejecutados utilizando los mismos parámetros, indicados en la Tabla II. La escala de las bases de datos y la elección de los parámetros de cada método se realizan siguiendo las indicaciones dadas en [4].

Distinguiremos dos modelos de particiones usados en este trabajo:

- *Validación cruzada en 10* folds *clásica (Tfcv clásica)*: en donde T_i, $i = 1, ..., 10$ es un 90 % de D y TS_i su complementario 10 % de D. Se obtiene tal y como

Class	Instancias
A	{1,2,3,4,5,6,7}
B	{8,9,10,11,12,13}

	Solución Actual	→	Solución Vecina
Representación	0110110100010	→	01**0**0110100010
Estructura U	$\{3, 5, 8, 8, 3, 2, 6, 2, 8, 8, 8, 3, 2, 3\}$	→	$\{\mathbf{12}, 5, 8, 8, \mathbf{2}, 2, 6, 2, 8, 8, 8, \mathbf{8}, 2, \mathbf{8}\}$
Aportación al Fitness	{1,1,0,0,1,1,1,0,1,1,0,0,0}	→	{-1,·,·,·,0,·,·,·,·,·,+1,·,+1}
Fitness	7	→	$7 - 1 + 1 + 1$
Cuenta de Evaluación Parcial: EP = $\frac{N_{nu}}{\lvert T\rvert} = \frac{4}{13}$			
Fitness del Vecino: 8			

Fig. 4. Ejemplo de un movimiento en el procedimiento meme y evaluación parcial

TABLA I
CARACTERÍSTICAS PRINCIPALES DE LOS CONJUNTOS DE DATOS

Nombre	N° Ejemplos	N° Atributos	N° Clases
Adult	45222	14	2
Nursery	12960	8	5
Page-Blocks	5476	10	5
Pen-Based	10992	16	10
Satimage	6435	36	7
SpamBase	4597	57	2
Splice	3190	60	3
Thyroid	7200	21	3

TABLA II
PARÁMETROS CONSIDERADOS DE LOS ALGORITMOS

Algoritmo	Parámetros
CHC	$Pop = 50, Eval = 10000, \alpha = 0{,}5$
IGA	$Pop = 10, Eval = 10000$ $p_m = 0{,}01, \alpha = 0{,}5$
GGA	$P_m = 0{,}001, P_c = 0{,}6, Pop = 50$ $Eval = 10000, \alpha = 0{,}5$
PBIL	$LR = 0{,}1, Mut_{shift} = 0{,}05, p_m = 0{,}02, Pop = 50$ $Negative_{LR} = 0{,}075, Eval = 10000$
SSGA	$P_m = 0{,}001, P_c = 1, Pop = 50$ $Eval = 10000, \alpha = 0{,}5$
SSMA	$Pop = 10, Eval = 10000, p_m = 0{,}01, p_c = 1$

indican las ecuaciones 4 y 5.

$$TR_i = \bigcup_{j \in J} D_j, J = \{j/1 \leq j \leq (i-1) \\ and\ (i+1) \leq j \leq 10\}, \quad (4)$$

$$TS_i = D \setminus TR_i. \quad (5)$$

▪ *Validación cruzada en 10* folds *estratificada (Tfcv strat)*: en donde $SPSS_i$ es generado usando DS_j en vez de D_j (ver [5]).

$$SPSS_i = \bigcup_{j \in J} DS_j, J = \{j/1 \leq j \leq b \cdot (i-1) \\ and\ (i \cdot b) + 1 \leq j \leq t\} \quad (6)$$

Los conjuntos de datos considerados son particionados usando *Tfcv clásica* (ver expresiones 4 y 5 excepto para el conjunto *Adult*, el cual es particionado usando el procedimiento *Tfcv strat* con $t = 10$ y $b = 1$ (ver expresión 6).

Para comparar resultados, consideramos el uso de tests no paramétricos de acuerdo a las recomendaciones hechas en [7]. Estos son más seguros que los tests paramétricos porque no asumen la existencia de la distribución normal o la homogeneidad de varianza. Como consecuencia, los tests no paramétricos pueden ser aplicados a aciertos en clasificación, ratios de error o cualquier otra medida para evaluación de clasificadores, incluyendo incluso modelos de tamaño y tiempo de cómputo. Los resultados empíricos nos sugieren que son más robustos que los proporcionados por los paramétricos. Demšar recomienda un conjunto de test no paramétricos simple, seguro y robusto para efectuar comparaciones estadísticas de clasificadores, uno de ellos es el *Wilcoxon Signed-Ranks Test* [23].

Vamos a presentar dos tipos de tablas, cada una de acuerdo a la siguiente estructura:

1. *Tabla Completa de Resultados*: Muestra la media de los resultados obtenidos por cada algoritmo en los conjuntos de datos evaluados:
   - La primera columna muestra el nombre del algoritmo.
   - La segunda columna contiene el tiempo medio de ejecución asociado al algoritmo. Han sido ejecutados en un HP Proliant DL360 G4p, Intel Xeon 3.0 Ghz, 1 Gb RAM.
   - La tercera columna muestra el porcentaje medio de reducción a partir del conjunto inicial de entrenamiento.
   - La cuarta y quinta columna incluye la media de acierto cuando 1-NN es aplicado usando S; la primera muestra el acierto de entrenamiento y la segunda muestra el acierto de test al clasificar el conjunto test con S.

   En la tercera, cuarta y quinta columna, los mejores resultados por columna son mostrados en negrita.
2. *Tablas del Test de Wilcoxon para n x n Comparaciones*: Debido a que la evaluación de justo la media del acierto de clasificación sobre todos los conjuntos de datos esconde información importante y cada conjunto de datos representa y problema diferente de clasificación con distintos grados de dificultad, hemos incluido un segundo tipo de tabla realizando una comparación estadística de métodos sobre múlti-

TABLA III
RESULTADOS MEDIOS DE LOS ALGORITMOS DE SPE

Algoritmo	**Tiempo(s)**	**% Red.**	**% Ac. Trn**	**% Ac. Test**
CHC	4357,56	98,84	86,53	85,75
GGA	16287,52	90,37	88,36	87,03
IGA	41319,74	70,15	90,25	87,44
PBIL	13303,32	83,41	89,99	87,86
SSGA	11161,94	93,83	90,11	87,55
SSMA	2288,14	**99,05**	**90,59**	**88,46**

ples conjuntos de datos. Tal y como hemos mencionado, Demšar recomienda un conjunto de test no paramétricos simple, seguro y robusto para efectuar comparaciones estadísticas de clasificadores, uno de ellos es el *Wilcoxon Signed-Ranks Test* [23].
Una tabla de Wilcoxon, en esta sección, se divide en dos partes: En la primera parte, efectuamos el test de Wilcoxon usando como medida de rendimiento el acierto en clasificación en el conjunto de test y en la segunda parte, usamos un balance entre reducción y acierto en clasificación. Este balance corresponde a $0{,}5 \cdot tasa\_clas + 0{,}5 \cdot porc\_red$. La estructura de las tablas presenta $N_{alg} \times N_{alg}+2$ celdas para comparar todos los algoritmos entre ellos. En cada una de las $N_{alg} \times N_{alg}$ pueden aparecer tres símbolos: +, - ó =. Éstos indican que el algoritmo situado en la fila es mejor (+), peor (-) o igual (=) en comportamiento (acierto o el balance definido anteriormente) que el algoritmo que aparece en la columna. Esta determinación de diferencias se realiza considerando un nivel de significancia del test $p = 0,05$. La penúltima columna representa el número de algoritmos con peor o igual comportamiento que el que aparece en la fila (sin considerar el algoritmo propio) y en la última columna, se representa el número de algoritmos con peor comportamiento que el que aparece en la fila.

La Tabla III muestra los resultados medios para los algoritmos de SPE ejecutados los conjuntos de datos señalados en la Tabla I. La Tabla IV muestra las diferencias estadísticas usando el test de Wilcoxon entre los algoritmos de SPE de acuerdo a considerar el rendimiento en acierto por un lado, y el balance acierto-reducción por otro.

Observando las Tablas III y IV, se pueden destacar los siguientes aspectos:
▪ En la Tabla III, *SSMA* presenta la mejor tasa de reducción y acierto, tanto en entrenamiento como en test.
▪ El tiempo de ejecución también se reduce de forma considerable; con respecto al siguiente modelo más rápido (*CHC*), alcanza una tasa del 50 % del tiempo empleado por éste, y si comparamos con el modelo estacionario clásico (*SSGA*), el tiempo que emplea se reduce hasta casi cinco veces más.
▪ Como puede observarse en la Tabla IV, todos los algoritmos de SPE son muy competitivos considerando solo acierto en clasificación. Sin embargo, esto no ocurre cuando también interviene la tasa de reducción en el rendimiento. Es este caso, los resultados del test de Wilcoxon indican que *SSMA* supera al resto de modelos de SPE.

TABLA IV
TABLA DEL TEST DE WILCOXON

Acierto en Test								
	6	*5*	*4*	*3*	*2*	*1*	>=	>
CHC (1)	-	=	=	=	=		4	0
GGA (2)	-	-	-	=		=	2	0
IGA (3)	=	=	=		=	=	5	0
PBIL (4)	=	=		=	+	=	5	1
SSGA (5)	=		=	=	+	=	5	1
SSMA (6)		=	=	=	+	+	5	2
Balance Acierto-Reducción								
	6	*5*	*4*	*3*	*2*	*1*	>=	>
CHC (1)	-	=	+	+	+		4	3
GGA (2)	-	-	+	+		-	2	2
IGA (3)	-	-	-		-	-	0	0
PBIL (4)	-	-		+	-	-	1	1
SSGA (5)	-		+	+	+	=	4	3
SSMA (6)		+	+	+	+	+	5	5

• Los resultados nos permiten distinguir tres tipos de AEs a partir del grado de logro de cada uno de los objetivos:
• Métodos que no convergen adecuadamente para conseguir una alta tasa de acierto, pero que consiguen excelentes tasas de reducción: *CHC* y *GGA*.
• Métodos que alcanzan porcentajes de acierto adecuados pero que descuidan la reducción del conjunto seleccionado: *IGA* y *PBIL*.
• Métodos que se centran en lograr optimizar ambos objetivos de forma adecuada: *SSGA* y *SSMA*. Curiosamente, ambos son modelos evolutivos estacionarios.

V. CONCLUSIONES

En este trabajo se presenta un nuevo modelo de algoritmo evolutivo para el problema de la Selección de Prototipos. Se trata de un Algoritmo Memético que incorpora una búsqueda local pensada para este problema.

Hemos desarrollado un estudio experimental estableciendo una comparación entre nuestra propuesta y los métodos evolutivos anteriores estudiados en la literatura. Las principales conclusiones alcanzadas son:
▪ Nuestra propuesta de *SSMA* presenta la mejor tasa de reducción y acierto.

- *SSMA* consigue obtener el menor tiempo de cómputo con respecto a los otros modelos de SPE. El aumento de eficiencia es muy significativo porque reduce el tiempo de la propuesta más rápida a un 50 %, y de la siguiente más rápida a un 20 %.
- El test basado en rankings con signos de Wilcoxon nos informa de la presencia de alta competitividad entre todas las propuestas de SPE considerando únicamente la eficacia en clasificación.
- Por otro lado, el test Wilcoxon obtiene diferencias entre todos los métodos cuando también consideramos como objetivo la capacidad de reducción. Si estableciéramos un orden de calidad, la propuesta *SSMA* quedaría en el puesto más alto.
- Los modelos evolutivos estacionarios se comportan mejor que el resto de modelos evolutivos para el problema de la Selección de Prototipos.

Agradecimientos

Este trabajo ha sido subvencionado por el Ministerio de Educación Español, en el marco de los proyectos TIN2005-08386-C05-01 y TIN2005-08386-C05-03.

Referencias

[1] D.W. Aha, D. Kibler, M. Albert, *Instance-based learning algorithms*, Machine Learning, 6, 37-66, 1991.

[2] S. Baluja, *Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning*, Technical Report CMU-CS-94-163, Pittsburgh, PA, 1994.

[3] J. C. Bezdek, L. Kuncheva, *Nearest prototype classifier designs: An experimental study.*, Int. J. Intelligent Systems, 16(12), 1445-1473, 2001.

[4] J.R. Cano, F. Herrera, M. Lozano, *Using evolutionary algorithms as instance selection for data reduction in KDD: an experimental study*, IEEE Transactions on Evolutionary Computation, 7(6), 561-575, 2003.

[5] J.R. Cano, F. Herrera, M. Lozano, *Stratification for Scaling Up Evolutionary Prototype Selection*, Pattern Recognition Letters, 26(7), 953-963, 2005.

[6] R. Dawkins, *The Selfish Gene*, Oxford University Press, 1990.

[7] J. Demšar, *Statistical comparisons of classifiers over multiple data sets*, Journal of Machine Learning Research, 7, 1-30, 2006.

[8] A.E. Eiben, J.E. Smith, *Introduction to Evolutionary Computing*, Springer Verlag, 2003.

[9] L.J. Eshelman, *The CHC adaptative search algorithm: How to have safe search when engaging in nontraditional genetic recombination*, Eds: G.J.E. Rawlins, In: Foundations of Genetic Algorithms, 261-283, 1991.

[10] M. Grochowski, N. Jankowski, *Comparison of Instance Selection Algorithms II. Results and Comments*, ICAISC, LNCS 3070, 580-585, 2004.

[11] D.E. Goldberg, K. Deb, *A comparative analysis of selection schemes used in genetic algorithms*, Eds: G.J.E. Rawlins, In: Foundations of Genetic Algorithms, 69-92, 1991.

[12] S. Ho, C. Liu, S. Liu, *Design of an optimal nearest neighbor classifier using an intelligent genetic algorithm*, Pattern Recognition Letters, 23(13), 1495-1503, 2002.

[13] N. Krasnogor, J. Smith, *A Tutorial for Competent Memetic Algorithms: Model, Taxonomy, and Design Issues*, IEEE Transactions on Evolutionary Computation, 9(5), 474-488, 2005.

[14] M.W.S. Land, *Evolutionary Algorithms with Local Search for Combinatorial Optimization*, Phd. Thesis Dissertation, University of California, San Diego, 1998.

[15] Y. Leung, Y. Wan, *An orthogonal genetic algorithm with quantization for global numerical optimization*, IEEE Transactions on Evolutionary Computation, 5(1), 41-53, 2001.

[16] M. Lindenbaum, S. Markovitch, D. Rusakov, *Selective Sampling for Nearest Neighbor Classifiers*, Machine Learning, 54(2), 125-152, 2004.

[17] M. Lozano, F. Herrera, N. Krasnogor, D. Molina, *Real-coded memetic algorithms with crossover hill-climbing*, Evolutionary Computation, 12(3), 273-302, 2004.

[18] D.J. Newman, S. Hettich, C.L. Blake, C.J. Merz, *UCI Repository of machine learning databases*, http://www.ics.uci.edu/∼mlearn/MLRepository.html, 1998.

[19] P. Moscato, *On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms*, Technical Report C3P 826, Pasadena, CA, 1989.

[20] A. Papadopoulos, *Nearest Neighbor Search: A Database Perspective*, Springer Verlag, 2004.

[21] R. Paredes, E. Vidal, *Learning Weighted Metrics to Minimize Nearest-Neighbor Classification Error*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(7), 1100-1110, 2006.

[22] G. Shakhnarovich, T. Darrel, P. Indyk, *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*, MIT Press, 2006.

[23] D.J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, CRC Press, 2000.

[24] H. Wang, *Nearest Neighbors by Neighborhood Counting*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(6), 942-953, 2006.

[25] D.R. Wilson, T.R. Martinez, *Reduction Techniques for Instance-Based Learning Algorithms*, Machine Learning, 38(3), 257-286, 2000.