

Mérida-Almendralejo  
4 al 6 de Febrero de 2015

15

X Congreso Español sobre Metaheurísticas,  
Algoritmos Evolutivos y Bioinspirados  
MAEB 2015

# Índice General

## Sesión general

<b>Planificación de celdas de reporte con el algoritmo SPEA2</b> <i>Víctor Berrocal-Plaza, Miguel A. Vega-Rodríguez, Juan M. Sánchez-Pérez .....</i>	<b>1</b>
<b>Algoritmo Genético con Diversificación Voraz y Equilibrio entre Exploración y Explotación</b> <i>Andrés Herrera-Poyatos, Francisco Herrera .....</i>	<b>9</b>
<b>Introducing Mixtures of Generalized Mallows in Estimation of Distribution Algorithms</b> <i>Josian Santamaría, Josu Ceberio, Alexander Mendiburu, Roberto Santana, José A. Lozano .....</i>	<b>19</b>
<b>Un Algoritmo Evolutivo para la Reducción de Tiempos de Viaje y Emisiones Utilizando Paneles LED</b> <i>Daniel H. Stolfi, Enrique Alba .....</i>	<b>27</b>
<b>Algoritmo memético basado en regiones con archivo externo para optimización multimodal</b> <i>Benjamin Lacroix, Daniel Molina, Francisco Herrera .....</i>	<b>35</b>
<b>El problema de los cortafuegos. Resultados con métodos heurísticos y con programación lineal entera</b> <i>Carlos García-Martínez, Christian Blum, Francisco Javier Rodríguez, Manuel Lozano .....</i>	<b>43</b>
<b>Estudio preliminar sobre visualización y clasificación de la calidad de la emisión de sonido en el Clarinete</b> <i>Francisco Fernández de Vega, Francisco Chávez de La O, Carlos M. Fernandes, Antonio Mora, J.J. Merelo ....</i>	<b>51</b>
<b>Generación de Secuencias de Pruebas Funcionales con Algoritmos Bio-inspirados</b> <i>Javier Ferrer, Peter M. Kruse, Francisco Chicano, Enrique Alba .....</i>	<b>59</b>
<b>Ajuste Probabilístico de Modelos de Glucosa obtenidos mediante Gramáticas Evolutivas</b> <i>J. Ignacio Hidalgo, Rafael Villanueva, José Manuel Colmenar, José L. Risco-Martín, Esther Maqueda, Juan Carlos Cortés, Almudena Sánchez, Marta Botella, José Antonio Rubio, Juan Lanchares, Óscar Garnica, Alfredo Cuesta, Francisco Santonja, Iván Contreras, José Manuel Velasco .....</i>	<b>67</b>
<b>Una Metaheurística Multiarranque para el Problema de la Partición Entera Común Mínima</b> <i>Manuel Lozano, Francisco Javier Rodríguez, Carlos García-Martínez .....</i>	<b>75</b>
<b>Beam Search para la búsqueda de caminos en redes complejas con entidades semánticas</b> <i>Francisco Vélez, Enrique Herrera-Viedma, Óscar Cordón .....</i>	<b>83</b>
<b>Registrado evolutivo de fragmentos craneales en 3D mediante Scatter Search</b> <i>Enrique Bermejo, Alejandro León, Sergio Damas, Óscar Cordón .....</i>	<b>91</b>
<b>A Comparison of Estimation of Distribution Algorithms for the Linear Ordering Problem</b> <i>Josu Ceberio, Alexander Mendiburu, José A. Lozano .....</i>	<b>97</b>

# Algoritmo Genético con Diversificación Voraz y Equilibrio entre Exploración y Explotación

A. Herrera-Poyatos<sup>1</sup>, F. Herrera<sup>2</sup>

**Resumen--** Los algoritmos genéticos pueden presentar una rápida convergencia cuando la diversidad de la población no es suficiente para mantener equilibrio entre exploración y explotación de los operadores genéticos al generar nuevos cromosomas.

En este trabajo presentamos un algoritmo genético híbrido que pretende conseguir equilibrio entre diversidad y convergencia mediante la actualización de la población con nuevos individuos proporcionados por un algoritmo voraz aleatorizado y el uso de una competición entre padres e hijos para mantener un buen nivel de convergencia, junto con otras componentes específicas para el algoritmo.

**Palabras clave--** Algoritmos Genéticos, diversidad versus convergencia, problema del viajante de comercio

## I. INTRODUCCIÓN

La inteligencia artificial siempre ha intentado que las máquinas tengan un comportamiento similar al de los seres vivos. Los algoritmos bioinspirados [1] forman parte de la metáfora artificial del comportamiento de ciertas especies de entes biológicos. Los algoritmos bioinspirados son actualmente un área relevante dentro de la inteligencia artificial. Entre ellos podemos destacar los algoritmos genéticos (AGs) [2], [3] o los algoritmos de optimización basados en colonias de hormigas [4].

Centrándonos en los primeros, los AGs se basan en la aplicación de un conjunto de operadores, inspirados en los conceptos de la evolución natural y la genética, sobre una población de soluciones a un problema, llamadas cromosomas, hasta satisfacer un criterio de parada asociado a la calidad de la solución o el tiempo de respuesta. Son ampliamente utilizados en todas las áreas de las ciencias experimentales, salud e ingeniería permitiendo resolver múltiples problemas de optimización y ofreciendo ciertas ventajas frente a sus competidores.

La diversidad de la población es sin duda una de las piedras angulares sobre las que gira el buen funcionamiento de los AGs. Supongamos que durante un intervalo de tiempo se mantiene el proceso de evolución sobre la población pero no se encuentra una mejora evolutiva clara. En tal caso todos los individuos tienden a parecerse al mejor cromosoma obtenido y se dice que el algoritmo converge a un óptimo, en algunos casos a un óptimo local por la falta de diversidad de la población. El problema del equilibrio entre diversidad y convergencia es recurrente en la literatura especializada, donde se pueden encontrar múltiples propuestas que lo abordan desde diferentes perspectivas [5], [6].

En este trabajo abordamos el problema de la diversidad presentando un AG híbrido que permite añadir diversidad a la población en caso de que sea necesario, al mismo tiempo que mantiene la presión para potenciar la explotación de las zonas óptimas del espacio de soluciones. Para obtener la mencionada diversidad se utiliza el procedimiento *diversificar* basado en los algoritmos voraces aleatorizados [7] y el concepto de hibridación [8],[9]. Consiste en la sustitución de aquellas soluciones que verifiquen determinado criterio de semejanza con otras de la población por nuevas soluciones generadas por estos algoritmos atendiendo a la lista de candidatos y dando lugar a diferente material genético en el proceso. La convergencia se consigue mediante la aplicación de la competición entre padres e hijos, introducido en los algoritmos de evolución diferencial [10]. Esto se complementa con un modelo específico de selección de padres que permite alcanzar el mencionado equilibrio y que denominamos selección aleatoria adyacente. Nos referiremos al algoritmo propuesto como Algoritmo Genético Equilibrado con Diversificación Voraz (AGEDV).

Mostraremos los resultados experimentales de la nueva propuesta así como la efectividad de la misma para mantener la diversidad de la población y la convergencia. Para ello utilizaremos como caso de estudio el conocido problema del viajante de comercio [11], ampliamente empleado en la literatura especializada para el estudio de algoritmos de optimización.

<sup>1</sup> E.T.S. de Ingeniería Informática, Universidad de Granada  
E-mail: andreshp9@gmail.com

<sup>2</sup> Departamento de Ciencias de la Computación e Inteligencia Artificial, E.T.S. de Ing. Informática, Universidad de Granada  
E-mail: herrera@decsai.ugr.es

El trabajo se organiza en las siguientes secciones. En la sección II profundizaremos en el concepto de AG. En la sección III introduciremos el concepto de diversidad de la población y presentaremos el algoritmo AGEDV. En la sección IV se analizan los resultados experimentales realizados sobre el problema del viajante de comercio desde una triple perspectiva, efectividad, convergencia y diversidad de la población. Mostraremos cómo afecta cada componente del algoritmo en alcanzar el equilibrio entre diversidad y convergencia y se compara el algoritmo con otros AGs que buscan el mencionado equilibrio. En la última sección desarrollaremos las conclusiones obtenidas y se explicarán las vías de trabajo futuro.

## II. ALGORITMOS GENÉTICOS

En esta sección presentamos una breve introducción a los AGs. En la primera subsección se realiza una descripción de los mismos y se proporciona el pseudocódigo del algoritmo utilizado. En la segunda se estudia la aplicación de los AGs al problema del viajante de comercio.

### A. Descripción de los algoritmos genéticos

Consideremos un problema de optimización. Sea  $S$  el conjunto de todas las soluciones al problema y  $f$  una función real definida sobre  $S$ ,  $f: S \rightarrow \mathbb{R}$ , tal que dada la solución del problema  $s$  nos proporciona una medida de la calidad de la solución,  $f(s)$  o valor de la función objetivo en  $s$ . Cuanto menor sea  $f(s)$  mejor es la solución si nos encontramos con un problema de minimización y viceversa si nos encontramos con un problema de maximización. Supongamos sin pérdida de la generalidad que nos encontramos en el primer caso. Deseamos encontrar una solución  $s'$  tal que minimice  $f$ . En tal contexto, consideramos un conjunto de soluciones que conforma la población del AG y que varía en función del tiempo  $t$ . A la población en el tiempo o iteración  $t$  se la denota como  $P(t)$ . Con todo lo anterior, podemos definir un AG como una heurística probabilística tal que dada  $P(t)$  utiliza el proceso de selección, y los operadores genéticos de cruce y mutación para obtener  $P(t+1)$ .

Definimos para lo que sigue una aplicación  $MS: \mathbb{N} \rightarrow S$  que nos proporcione para un tiempo  $t \in \mathbb{N}$  la mejor solución de la población  $P(t)$  en términos de  $f$ . Análogamente, definimos la aplicación  $PS: \mathbb{N} \rightarrow S$  que nos proporciona la peor solución de  $P(t)$ .

Un AG con la definición anterior no garantiza que al finalizar el proceso se hayan obtenido mejores soluciones a lo largo de la evolución. Esto sí se cumple si además le imponemos la siguiente desigualdad:

$$f(MS(t)) \leq f(MS(t-1)) \quad \forall t \in \mathbb{N}$$

El cumplimiento de la desigualdad anterior implica que la mejor solución de la población  $P(t)$  es mejor o igual que las soluciones de las poblaciones predecesoras. Para conseguir su verificación aplicamos el criterio de elitismo, sustituyendo la peor solución en  $P(t)$  por la mejor de  $P(t-1)$ . Un AG sin elitismo no suele converger a la solución óptima del problema [12]. Por consiguiente, el elitismo es un eslabón básico para el buen funcionamiento de los GAs.

En este trabajo consideramos un modelo generacional con elitismo en el que para cada iteración del AG, dada  $P(t)$ , se genera una nueva población  $P(t+1)$  mediante los operadores genéticos tal y como indica el siguiente pseudocódigo:

```
Función crearNuevaGeneración(  $P(t)$  )
 $P(t+1) \leftarrow \emptyset;$ 
While ( $\text{size}(P(t+1)) < \text{size}(P(t))$ ) do
     $padre1 \leftarrow \text{seleccion}(P(t));$ 
     $padre2 \leftarrow \text{seleccion}(P(t));$ 
    if ( $\text{realEntre}0y1() < \text{prob\_cruce}$ )
         $hijo1 \leftarrow \text{cruce}(padre1, padre2);$ 
         $hijo2 \leftarrow \text{cruce}(padre2, padre1);$ 
    else
         $hijo1 \leftarrow padre1;$ 
         $hijo2 \leftarrow padre2;$ 
    endif
    if ( $\text{realEntre}0y1() < \text{prob\_mutacion}$ )
         $\text{mutacion}(hijo1);$ 
    endif
    if ( $\text{realEntre}0y1() < \text{prob\_mutacion}$ )
         $\text{mutacion}(hijo2);$ 
    endif
     $P(t+1) \leftarrow P(t+1) \cup \{hijo1, hijo2\};$ 
endwhile
 $PS(t+1) \leftarrow MS(t);$ 
return  $P(t+1);$ 
```

Nos encontramos las variables  $\text{prob\_cruce}$  y  $\text{prob\_mutacion}$  que indican la probabilidad de que se ejecute el operador de cruce o de mutación respectivamente. Utilizamos los valores usuales  $\text{prob\_cruce} = 0.7$  y  $\text{prob\_mutacion} = 0.1$  por cromosoma. El tamaño de la población del AG se mantendrá en 60 cromosomas.

A partir de esta función se obtiene de manera directa el AG:

```
Algoritmo Genético Generacional
 $t \leftarrow 0;$ 
 $\text{inicializar } P(t);$ 
While (Not CriterioParada) do
     $t \leftarrow t + 1;$ 
     $P(t) \leftarrow \text{crearNuevaGeneración}(P(t-1));$ 
endwhile
```

### B. Aplicación al Viajante de Comercio

Como aplicación con la que se realiza el estudio experimental se ha empleado el famoso problema del viajante de comercio [11] (conocido por la abreviación de su nombre en inglés, TSP). Consiste en, dado un grafo completo y ponderado, obtener el ciclo Hamiltoniano que minimice la suma de las ponderaciones de los arcos que lo constituyen. A tal suma se la llama coste de la solución. Así pues, la función  $f$  lleva cada solución a su coste correspondiente. La elección de este problema se debe a la existencia de un algoritmo voraz aleatorizado en el mismo y a que gracias a su enunciado intuitivo resulta sencillo representar la información obtenida, siendo además un problema utilizado con frecuencia en el estudio de los AGs.

Se han desarrollado numerosos operadores genéticos para el viajante de comercio. Como operador de selección se suele utilizar la selección por torneo, en la cual se toman  $k$  individuos de la población y se devuelve el mejor en términos de  $f$ . Tenemos por tanto un parámetro  $k$  variable que mantenemos en  $k=2$ . Por otro lado, implementamos el conocido operador de cruce OX y la mutación por intercambio [13]. La población se inicializa de manera aleatoria como es usual en la literatura.

### III. ALGORITMO GENÉTICO EQUILIBRADO CON DIVERSIFICACIÓN VORAZ

La sección actual se divide en tres subsecciones. En la primera se introduce el concepto de diversidad de la población del AG. En la segunda subsección se desarrolla el mecanismo de diversificación en el que se basa nuestra propuesta. En tercer lugar se presenta el algoritmo AGEDV y se razona su funcionamiento.

#### A. Diversidad en la población de los AGs:

Consideremos un problema de optimización con un espacio de soluciones  $S$ . Supongamos que se dispone de una medida de la distancia entre dos soluciones,  $d: S \times S \rightarrow \mathbb{R}^+ \cup \{0\}$ , de forma que  $s = s'$  si, y solo si,  $d(s, s') = 0$ , verificando, además, las correspondientes propiedades matemáticas. Sea  $n$  el número de cromosomas que componen la población, se define la diversidad de  $P(t)$  como la media de la distancia media de cada solución a todas las demás:

$$D(t) = \frac{\sum_{s, s' \in P(t)} d(s, s')}{(n-1)n}$$

En el caso del viajante de comercio, tomamos como medida  $d$  aquella que nos proporciona el número de arcos en los que difieren dos soluciones. Es claro que la máxima distancia posible entre dos soluciones responde al número de ciudades del problema en cuestión.

Si realizamos una ejecución del AG podemos observar que la diversidad evoluciona como nos muestra la Figura 1, tomada sobre la instancia berlin52 [14]. Comienza en un valor cercano al máximo posible ya que las soluciones aleatorias son muy diferentes entre sí. La diversidad disminuye rápidamente al centrarse el algoritmo en una zona del espacio de soluciones. Sin embargo, esta disminución es excesiva pues la diversidad oscila hasta estabilizarse en un valor cercano al 0 cuando la población ha convergido a un óptimo local. Nuestro propósito es evitar este hecho para mejorar ampliamente el rendimiento del algoritmo.

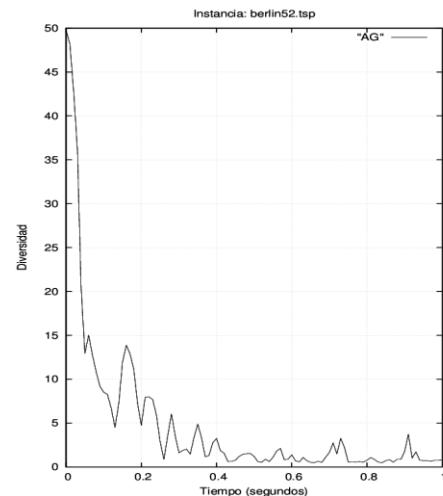


Fig. 1. Diversidad del algoritmo genético

La diversidad en la población del AG es en parte consecuencia del operador de mutación, dependiendo de la probabilidad otorgada al mismo. De ser esta 0, tras varias iteraciones la diversidad tiende a ser nula, estancándose el algoritmo en un óptimo local. Si aumentamos tal probabilidad, el valor en torno al cual se estabiliza la diversidad aumenta. Sin embargo, el operador de mutación proporciona diversidad a costa de empeorar de manera genérica la calidad de las soluciones. Por ello, la probabilidad de mutación se fija en valores bajos en la literatura especializada (entre 0.1 y 0.2 por cromosoma), no permitiendo una alta diversidad como se muestra en la Figura 1 (donde la probabilidad de mutación es 0.1).

En este trabajo tratamos de proporcionar un mecanismo distinto que aumente la diversidad de la población de manera apropiada sin dar lugar a una pérdida de calidad de las soluciones en el proceso. Además, debe potenciar el buen funcionamiento de los operadores genéticos.

#### B. Diversificación Voraz de la Población

La diversidad en la población por sí misma es un arma de doble filo. Es necesaria para llevar a cabo el proceso de exploración del espacio de soluciones

pero puede conllevar la no finalización del mismo, no consiguiendo realizar el proceso de explotación necesario para optimizar la mejor solución obtenida. Es deseable pues un mecanismo de diversificación progresivo que introduzca diversidad sólo cuando sea necesario.

Por otra parte, parece claro que no interesa que la población contenga cromosomas repetidos ya que este hecho disminuye la diversidad de la misma. Además, deseamos que los cromosomas que la constituyen sean de calidad en un doble sentido, es decir, no solo tengan un buen valor de la función objetivo sino que también sean potencialmente buenos para el operador de cruce.

Con esto en mente deseamos diseñar un procedimiento *diversificar* que sustituya los cromosomas que comparten determinadas características de similitud por nuevo material genético, siguiendo las pautas mencionadas, con el fin de obtener el siguiente algoritmo:

#### Algoritmo Genético Generacional con Diversificación

```

 $t \leftarrow 0;$ 
 inicializar P(t);
While (Not CondiciónFinal) do
     $t \leftarrow t + 1;$ 
     $P(t) \leftarrow crearNuevaGeneración(P(t-1));$ 
     diversificar(P(t));
endwhile
```

El procedimiento *diversificar* debe tener un coste computacional bajo puesto que la base del funcionamiento del algoritmo es el esquema evolutivo. Para obtener cromosomas bajo estas restricciones proponemos utilizar un algoritmo voraz aleatorizado [7]. Éste proporciona cromosomas aceptables desde el punto de vista de la función objetivo y que están elegidos a partir de aquel material genético que satisface determinado criterio de calidad gracias a la función de selección voraz. El hecho de que contenga cierta aleatoriedad en la elección de los cromosomas introduce la diversidad requerida en el proceso.

Un algoritmo voraz aleatorizado permite construir soluciones de los problemas que verifiquen que, dada una solución, esta se pueda representar como un conjunto de elementos y exista una función de selección que nos indique la calidad de determinado elemento en función de los que ya pertenezcan al conjunto solución. El proceso de construcción se mantiene durante una serie de pasos, en cada uno de los cuales se selecciona un elemento para la solución, hasta que esta se ha completado correctamente. En cada paso se crea una lista restringida de candidatos (LRC) que contiene los

mejores elementos a elegir con respecto de la función voraz correspondiente. La LRC puede tener un tamaño constante o, mejor, un tamaño variable y estar constituida por aquellos elementos que verifiquen ser, como máximo,  $(1+\sigma)$  veces peores en términos de la función voraz que el mejor elemento encontrado, donde  $\sigma$  es un valor real mayor que 0. Optamos por este segundo modelo pues permite una mayor calidad manteniendo diversidad entre las soluciones. Tras la creación de la LRC se selecciona un elemento aleatorio de la misma y se añade a la solución. Por último se adapta la función voraz a nuestra elección.

```

Función generarSoluciónVorazAleatorizada()
Solución  $\leftarrow \{\}$ ;
While (No esté construida la solución) do
    CrearLRC(LRC);
     $x \leftarrow seleccionarElementoAleatorio(LRC);$ 
    Solución  $\leftarrow$  Solución  $\cup \{x\};$ 
    adaptarFunciónVoraz(x);
endwhile
```

Particularizando en el problema del viajante de comercio, la función voraz asociada responde a la distancia de un nodo no elegido al último visitado, siendo éste mejor cuanto menor sea su distancia correspondiente. Así pues, la solución resultante constará de arcos relativamente cortos en su mayor parte. Como consecuencia, al combinar una solución voraz con otra dada mediante el operador de cruce se mantendrán gran parte de estos arcos, obteniendo un resultado de calidad.

Nuestra propuesta de procedimiento *diversificar* utiliza el algoritmo anterior para sustituir aquellos cromosomas que comparten determinadas características de similitud con otros de la población. De esta manera se aumenta la diversidad a la vez que se mantiene la calidad de los cromosomas. Consideramos determinada característica y el conjunto de sus posibles valores  $C$ . Definimos la aplicación  $g: S \rightarrow C$  tal que dada una solución  $s$  proporciona  $g(s)$ , valor de evaluación de la característica elegida para la solución  $s$ . Podemos definir así de manera genérica el procedimiento *diversificarVoraz*.

```

Procedimiento diversificarVoraz( $P(t), g(\cdot)$ ):
For  $s$  in  $P(t)$  do
    if ( $\exists s' \in P(t) \setminus \{s\}$   $t.q. g(s) = g(s')$ ) then
         $s \leftarrow$  generarSoluciónVorazAleatorizada();
    endif
endfor
```

Nótese que si varias soluciones comparten la característica evaluada por  $g$  siempre una de ellas permanecerá en la población.

La elección de  $g$  depende de la diversidad que se pretenda introducir. Un primer acercamiento consiste en utilizar la identidad en  $S$  ( $Id: S \rightarrow S$ ), sustituyendo aquellos cromosomas estrictamente repetidos en la población. La siguiente opción consiste en utilizar  $g=f$ , donde  $f$  es la función objetivo del problema, produciendo una sustitución a nivel de la función objetivo. En la sección 4 se comparan ambas propuestas.

### C. Algoritmo Genético Equilibrado con Diversificación Voraz (AGEDV)

El uso del mecanismo de diversificación produce de por sí una severa mejora en el AG clásico como veremos en la sección 4. Sin embargo, la sinergia entre este mecanismo y los demás operadores del algoritmo es ampliamente mejorable. Proponemos un nuevo modelo de AG con las siguientes características:

1. Mecanismo de selección de padres que potencie la diversidad en el cruce llamado selección aleatoria adyacente.
2. Probabilidad de cruce 1.
3. Se elimina el uso del operador de mutación.
4. Aplicación del concepto de competición entre padres e hijos para aumentar la presión ejercida sobre la población.
5. Diversificación voraz de la población en cada iteración del algoritmo.

Al nuevo algoritmo lo denominamos AGEDV pues consigue un buen equilibrio entre diversidad y convergencia utilizando la diversificación voraz. Exponemos a continuación con más profundidad sus componentes y el razonamiento tras las mismas.

Los mecanismos de selección tradicionales usados en los AGs tienden a ignorar las peores soluciones de la población asignándoles una baja probabilidad de elección. Esto le sucede por ejemplo a la selección por torneo o a la selección por ranking [15]. Si intentamos potenciar la diversidad de la población no podemos utilizar tales mecanismos pues estaremos deshaciéndonos de la diversidad introducida. Es más, deseamos que todo el material genético que conforma la población sea utilizado sistemáticamente en la operación de cruce para explorar el espacio de cromosomas y aprovechar las soluciones voraces obtenidas. Por ello proponemos ordenar aleatoriamente la población y cruzar las soluciones adyacentes (la primera y la última también se cruzan) generando una único hijo por pareja con 1 como probabilidad de cruce. A este mecanismo lo hemos denominado selección aleatoria adyacente. Con su uso se crea una nueva población  $P(t+1)$  de igual tamaño que la población anterior. Nótese que cada solución de  $P(t)$  ha sido padre de dos soluciones de  $P(t+1)$ , luego todo el

material genético de  $P(t)$  está latente en la nueva generación.

El proceso de selección anterior mantiene la diversidad de la población pero no introduce presión alguna. Recurrimos para ello al concepto de competición entre padres e hijos utilizado en los algoritmos de evolución diferencial [10]: cada hijo solo compite con su parente directo y el mejor de los dos es el que se mantendrá en  $P(t+1)$ . Como consecuencia, en la generación  $t+1$  siempre se encuentra para cada solución de  $P(t)$  o con ella misma o con un descendiente directo suyo, lo que garantiza que se mantenga su material genético. Además, la población  $P(t+1)$  siempre es mejor que su predecesora en términos de la función objetivo, el concepto de competición padres con hijos es un elitismo en sí mismo. Obtenemos la siguiente función para crear una nueva población:

```
Función crearNuevaPoblación(P(t))
P(t+1) ← Ø;
ordenarAleatoriamente(P(t));
for i in {0, 1, ..., size(P(t)) - 1} do
    padre1 ← elemento(P(t), i);
    padre2 ← elemento(P(t), (i+1) % size(P(t)));
    hijo ← cruce(padre1, padre2);
    if padre1 es mejor que hijo then
        P(t+1) ← P(t+1) ∪ padre1;
    else
        P(t+1) ← P(t+1) ∪ hijo;
    endif
endfor
return P(t+1);
```

El operador de mutación nos permitía explorar el espacio de vecinos de las soluciones de la población. Sin embargo, ya no es necesario para el correcto funcionamiento del algoritmo ya que su cometido es realizado por otros componentes del mismo: la selección aleatoria adyacente y el proceso *diversificarVoraz*. Presentamos a continuación el pseudocódigo del algoritmo AGEDV:

### Algoritmo Genético Equilibrado con Diversificación Voraz

```
t ← 0;
inicializar P(t);
While (Not CondiciónFinal) do
    t ← t + 1;
    P(t) ← crearNuevaPoblación(P(t-1));
    diversificarVoraz(P(t), Id(.));
endwhile
```

En la Figura 2 se muestra, para la instancia berlín52, la diversidad de la población en función del tiempo del algoritmo AGEDV en comparación con el AG, visto anteriormente. La diversidad inicial disminuye conforme ambos algoritmos se centran en

una zona del espacio de soluciones. Sin embargo, a partir de cierto punto, el algoritmo AGEDV mantiene la diversidad de la población en un alto valor gracias a la diversificación voraz y el funcionamiento del mismo, que permite trabajar con cromosomas de calidad en diferentes puntos del espacio de soluciones.

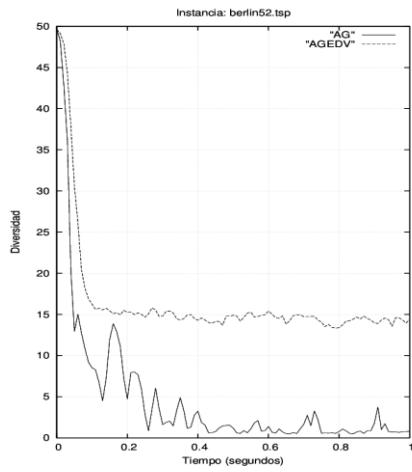


Fig. 2. El Diversidad del AGEDV y AG

#### IV. ANÁLISIS EXPERIMENTAL

Los experimentos se han realizados en un ordenador portátil de 8 GB de RAM y procesador Intel I5 a 2.5 GHz. Las 18 instancias del viajante de comercio se han obtenido de la biblioteca TSPLIB [14]. Todos los resultados presentan la media de 30 ejecuciones del correspondiente algoritmo.

La sección se divide en 3 subsecciones. En la primera se presenta un estudio sobre el tamaño de población para el AGEDV y el comportamiento del procedimiento *diversificar* en función de  $g$ . En la segunda subsección se compara el algoritmo con el AG clásico y otros modelos del estado del arte desde una triple perspectiva: calidad de las soluciones, convergencia al óptimo del problema y diversidad de la población. En la tercera subsección se analiza la aportación de las diferentes componentes del AGEDV a su funcionamiento.

##### A. Análisis de los parámetros del AGEDV

La Tabla I muestra la calidad media de las soluciones obtenidas en términos de la función objetivo así como su desviación típica para el algoritmo AGEDV con tamaños de población (TP) 30 y 60. Mantenemos el parámetro  $\sigma$  correspondiente al algoritmo voraz aleatorizado en 0.1 y se toma como función  $g$  la identidad en el espacio de soluciones ( $g=Id$ ). El tiempo de ejecución para cada instancia responde a  $0.1*N$  segundos, donde  $N$  es el número de ciudades de la misma. Señalamos en negrita la mejor media para cada problema. La última fila contiene el número de

instancias en las que el correspondiente tamaño de población ha conseguido el mejor / peor resultado.

TABLA I  
AGEDV CON  $g = Id$  Y TIEMPO = 0.1N SEGUNDOS

Problema	Óptimo	Calidad Media		Desviación Típica	
		AGEDV TP = 30	AGEDV TP = 60	AGEDV TP = 30	AGEDV TP = 60
eil51	426	<b>427.2</b>	427.267	<b>0.4</b>	1.26315
berlin52	7542	<b>7555.1</b>	7572.57	<b>39.3</b>	55.4068
st70	675	682.467	<b>682.067</b>	5.03808	<b>4.21057</b>
eil76	538	549.8	<b>549.5</b>	<b>0.97979</b>	1.43178
pr76	108159	<b>109169</b>	109395	942.768	<b>573.954</b>
kroA100	21282	21385.8	<b>21352.5</b>	134.201	<b>119.631</b>
rd100	7910	<b>7917.13</b>	7919.47	<b>28.619</b>	35.4445
eill01	629	<b>631.7</b>	633.3	<b>3.06757</b>	4.09186
lin105	14379	14436.2	<b>14430.5</b>	33.7544	<b>18.2916</b>
ch150	6528	6588.67	<b>6578.67</b>	<b>20.277</b>	20.6968
rat195	2323	2393.47	<b>2386.83</b>	<b>16.126</b>	17.6335
d198	15780	16076.4	<b>16053.9</b>	<b>75.0803</b>	95.1888
ts225	126643	127550	<b>127427</b>	<b>391.036</b>	524.795
a280	2579	2718.63	<b>2704.5</b>	<b>26.2824</b>	26.9923
lin318	42029	43815.3	<b>43739.5</b>	429.102	<b>412.694</b>
fl417	11861	12323.7	<b>12303.9</b>	<b>105.208</b>	134.313
pcb442	50778	<b>55741</b>	<b>55502</b>	<b>633.782</b>	639.032
rat575	6773	7677.5	<b>7670.97</b>	83.7412	<b>78.0105</b>
		<b>5 / 13</b>	<b>13 / 5</b>	<b>12 / 6</b>	<b>6 / 12</b>

El tamaño de población 60 obtiene mejores resultados a pesar de presentar una peor desviación típica, lo que indica que procesa el suficiente número de cromosomas como para encontrar probabilísticamente diferentes zonas de calidad en el espacio de soluciones. Mantenemos 60 como tamaño de población para el resto de la experimentación.

La Tabla II compara el funcionamiento del algoritmo AGEDV para el procedimiento *diversificar* a nivel de cromosomas ( $g=Id$ ) o a nivel de función objetivo ( $g=f$ ). El mejor funcionamiento de  $g=Id$  es claro,  $g=f$  introduce excesiva diversidad en la población al sustituir los cromosomas incluso antes de su completa repetición. También se muestra el porcentaje de soluciones que son generadas en la diversificación voraz, que oscila en torno al 2 y 5 %.

TABLA II  
AGEDV CON  $g = Id$  y  $g = f$ . TIEMPO = 0.1N SEGS.

Problema	Óptimo	Calidad Media		Porcentaje de Soluciones Voraces	
		AGEDV $g=Id$	AGEDV $g=f$	AGEDV $g=Id$	AGEDV $g=f$
eil51	426	<b>427.267</b>	428.4	5.36748	10.1318
berlin52	7542	<b>7572.57</b>	<b>7572.57</b>	5.08438	5.23617
st70	675	<b>682.067</b>	686.433	4.62881	7.36705
eil76	538	<b>549.5</b>	549.9	4.27079	8.90469
pr76	108159	<b>109395</b>	109503	4.55236	4.57542
kroA100	21282	<b>21352.5</b>	21384.6	4.90119	4.91509
rd100	7910	<b>7919.47</b>	7926.5	4.56272	4.70796
eill01	629	<b>633.3</b>	634.233	5.28149	7.87105
lin105	14379	14430.5	<b>14423.3</b>	4.82928	4.8212
ch150	6528	<b>6578.67</b>	6586.03	4.30821	4.49253
rat195	2323	<b>2386.83</b>	2400.87	3.88133	4.52266
d198	15780	<b>16053.9</b>	16076.2	4.18145	4.261
ts225	126643	127427	<b>127355</b>	2.7799	2.59045
a280	2579	<b>2704.5</b>	2727.53	4.09656	4.68289
lin318	42029	<b>43739.5</b>	43843.7	3.97038	4.03494
fl417	11861	12303.9	<b>12286</b>	4.01814	4.09853
pcb442	50778	55502	<b>55477</b>	2.72923	2.74129
rat575	6773	<b>7670.97</b>	7712.5	2.08298	2.21214
		<b>14 / 4</b>	<b>5 / 13</b>		

Cabe preguntarse si la diversidad aportada por el algoritmo voraz es la causa del comportamiento del algoritmo AGEDV o si bien el algoritmo funcionaría de igual manera introduciendo soluciones aleatorias, generando una mayor diversidad. Se ha realizado dicho estudio y los resultados muestran resultados similares al AG clásico y lejanos a los resultados del algoritmo AGEDV. Cómo se indica en la sección III, la causa puede estar en la calidad de los arcos de las soluciones generadas con el algoritmo voraz, arcos que aportan diversidad y calidad cuando se combinan mediante el operador de cruce. En la subsección B corroboramos este hecho incluso para otros algoritmos del estado del arte.

#### B. Comparación con algoritmos del estado del arte

En esta sección comparamos los resultados obtenidos con aquellos dados por el AG clásico y modelos de calidad contrastada que buscan añadir diversidad a la población por medios diferentes al operador de mutación: CHC [16] y Micro-AG [17].

CHC, presentado en 1990, es el primer AG en aplicar la competición entre padres e hijos. Adaptado al viajante de comercio [18], presenta las siguientes características:

- Tamaño de Población = 60
- Selección aleatoria con prevención de incesto para evitar el cruce de cromosomas similares.
- Competición entre padres e hijos: la población  $P(t+1)$  se crea eligiendo los mejores cromosomas entre padres e hijos.
- Reinicialización aleatoria en caso de convergencia de la población.

El micro-AG fue propuesto por Khrisnakumar en 1989 como AG con una población muy pequeña y rápida convergencia. El Micro-GA presenta las siguientes características:

- Tamaño de Población = 5
- El mejor individuo de  $P(t)$  se copia en  $P(t+1)$ .
- Variante de la selección por torneo para tomar dos pares de padres.
- Reinicialización aleatoria en caso de convergencia de la población (generalmente a igualdad de función objetivo).

En ambos modelos la probabilidad de cruce es 1 y se omite el operador de mutación, generando dos hijos por cada par de padres. La reinicialización aleatoria consiste en crear una nueva población tomando el mejor cromosoma de la anterior y los demás generados aleatoriamente.

En primer lugar presentamos otro argumento a favor de la hibridación con algoritmos voraces para

obtener diversidad: la reinicialización aleatoria de los dos algoritmos anteriores es muy inferior a una reinicialización que use un algoritmo voraz aleatorizado (reinicialización voraz). En la Tabla III se presenta una comparativa entre reinicialización aleatoria y reinicialización voraz para el CHC y micro-GA. Indiscutiblemente, el uso de los algoritmos voraces aleatorizados introduce una amplia mejora en ambos modelos.

TABLA III  
CHC Y MICRO-AG CON REINICIALIZACIÓN  
ALEATORIA Y VORAZ (TIEMPO = 0.1\*N)

Problema	Óptimo	Calidad Media			
		CHC	CHC R.Voraz	MicroGA	MicroGA R.Voraz
eil51	426	496.8	<b>443.933</b>	447.267	<b>432.267</b>
berlin52	7542	8041.8	<b>7633.1</b>	8053.27	<b>7588.1</b>
st70	675	889	<b>730.8</b>	743.967	<b>689.9</b>
eil76	538	665.2	<b>574</b>	586.933	<b>554.767</b>
pr76	108159	114084	<b>109572</b>	116626	<b>111017</b>
kroA100	21282	24010.4	<b>21377.7</b>	25627.2	<b>21689.2</b>
rd100	7910	9496.23	<b>7998.23</b>	9418.3	<b>8010.03</b>
eil101	629	827.7	<b>686.267</b>	729.433	<b>634.567</b>
lin105	14379	19445.4	<b>14426.4</b>	16827.7	<b>14511.4</b>
ch150	6528	9311.93	<b>6763.63</b>	9294.63	<b>6626.73</b>
rat195	2323	3515.8	<b>2431.4</b>	3566.53	<b>2430.53</b>
d198	15780	21395.6	<b>16603.7</b>	22493.2	<b>16387.3</b>
ts225	126643	214322	<b>133175</b>	251757	<b>129840</b>
a280	2579	5109.53	<b>2891.43</b>	5496.5	<b>2791.93</b>
lin318	42029	82239.7	<b>43917.3</b>	107959	<b>44886.4</b>
fl417	11861	32020.3	<b>12937</b>	60670.7	<b>12674.2</b>
pcb442	50778	117600	<b>57568.3</b>	174361	<b>58950.6</b>
rat575	6773	18170.7	<b>7773.43</b>	26568	<b>8024.4</b>

En la Tabla IV comparamos los algoritmos AG clásico, AGEDV, CHC y Micro-GA (utilizando la reinicialización voraz para los dos) desde el punto de vista de la calidad de las soluciones. El AGEDV obtiene el mejor resultado en 17 instancias de 18. La diversificación del algoritmo AGEDV permite adelantarse a la convergencia del mismo, lo que no sucede con la reinicialización del CHC y Micro-GA. (Subrayamos en cada caso la peor solución).

TABLA IV  
AG CLÁSICO, CHC, MICRO-GA Y AGEDV

Problema	Óptimo	Calidad Media			
		AG	CHC R.Voraz	MicroGA R. Voraz	AGEDV
eil51	426	441	<b>443.933</b>	432.267	<b>427.267</b>
berlin52	7542	<b>8123.03</b>	7633.1	7588.1	<b>7572.57</b>
st70	675	711.767	<b>730.8</b>	689.9	<b>682.067</b>
eil76	538	570.767	<b>574</b>	554.767	<b>549.5</b>
pr76	108159	<b>119699</b>	109572	111017	<b>109395</b>
kroA100	21282	<b>22769.7</b>	21377.7	21689.2	<b>21352.5</b>
rd100	7910	<b>8415.43</b>	7998.23	8010.03	<b>7919.47</b>
eil101	629	672.433	<b>686.267</b>	634.567	<b>633.3</b>
lin105	14379	<b>14888.2</b>	<b>14426.4</b>	14511.4	14430.5
ch150	6528	<b>6887.63</b>	6763.63	6626.73	<b>6578.67</b>
rat195	2323	<b>2524.67</b>	2431.4	2430.53	<b>2386.83</b>
d198	15780	<b>17235.3</b>	16603.7	16387.3	<b>16053.9</b>
ts225	126643	<b>135632</b>	133175	129840	<b>127427</b>
a280	2579	<b>2871.77</b>	2891.43	2791.93	<b>2704.5</b>
lin318	42029	<b>46927.4</b>	43917.3	44886.4	<b>43739.5</b>
fl417	11861	<b>13138.8</b>	12937	12674.2	<b>12303.9</b>
pcb442	50778	<b>58241.2</b>	57568.3	58950.6	<b>55502</b>
rat575	6773	<b>7877.9</b>	<b>7773.43</b>	8024.4	<b>7670.97</b>
		<b>0 / 11</b>	<b>1 / 5</b>	<b>0 / 2</b>	<b>17 / 0</b>

En la Tabla V se muestra la media del tiempo que tarda cada algoritmo en llegar a la solución óptima del problema en 30 ejecuciones de 20 segundos cada una. Si una ejecución finaliza sin encontrar el óptimo entonces el algoritmo presumiblemente se ha estancado en un óptimo local y no se utiliza para la media. A la derecha de la media del tiempo se muestra para cada instancia y algoritmo el número de veces que se ha alcanzado el óptimo del problema (NA si no se ha alcanzado en ninguna ejecución).

TABLA V  
CONVERGENCIA AL ÓPTIMO (SEGUNDOS / ÉXITOS)

Problema	Heurísticas			
	AG	CHC R. Voraz	MicroGA R. Voraz	AGEDV
berlin52	<u>15.76 / 1</u>	0.35 / <b>28</b>	0.67 / 23	<b>0.136 / 24</b>
kroA100	<u>NA</u>	12.576 / 8	10.77 / 6	<b>3.95 / 11</b>
rd100	<u>NA</u>	5.26 / 25	7.35 / 6	<b>3.81 / 30</b>

El nuevo modelo también presenta una mejor convergencia al óptimo del problema, ya que la diversidad añadida permite alcanzar el material genético pertinente para la obtención de la mejor solución posible.

Para finalizar el estudio de la convergencia, en la Figura 3 se puede apreciar cómo evolucionan todos los algoritmos durante una ejecución de 60 segundos sobre el problema d198. El AG clásico permanece más tiempo estancado en óptimos locales antes de salir de los mismos mientras que el nuevo modelo es el que mayor profundidad de búsqueda presenta.

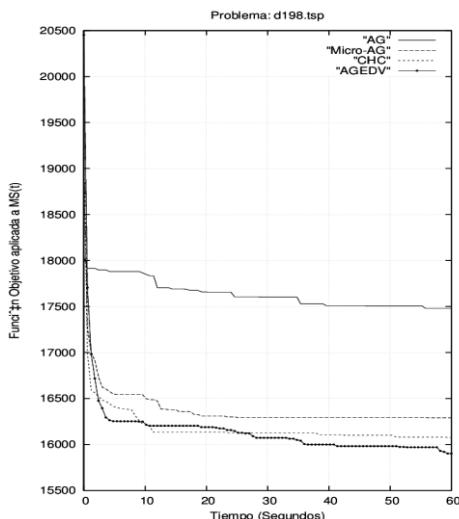


Fig. 3. Convergencia: AG vs Micro-AG con R. Voraz vs CHC con R. Voraz vs AGEDV

La Figura 4 presenta la evolución de la diversidad media en la ejecución anterior a partir del segundo 1. El AG no es capaz de mantener la diversidad en un nivel aceptable, hecho que si realizan los otros algoritmos gracias al uso de la reinicialización para introducir diversidad.

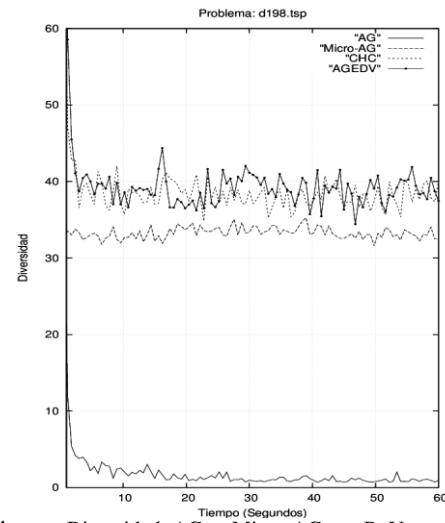


Fig. 4. Diversidad: AG vs Micro-AG con R. Voraz vs CHC con R. Voraz vs AGEDV

### C. Análisis de las componentes del AGEDV

La primera pregunta que se plantea es la influencia del proceso de diversificación voraz en el funcionamiento del algoritmo. En la Tabla II se mostró que en torno a un 5 % de las soluciones generadas por el AGEDV tienen lugar en este proceso. Para ver la importancia del mismo en la Tabla VI mostramos los resultados del algoritmo AGEDV sin diversificación voraz (AGE) comparado con el modelo con diversificación así como los resultados obtenidos al añadir la diversificación voraz al AG clásico (AG+DV).

TABLA VI  
ALGORITMOS CON O SIN DIVERSIFICACIÓN

Problema	Óptimo	Calidad Media			
		AG	AG+DV	AGEDV	AGE
eil51	426	441	433.2	<b>427.267</b>	768.9
berlin52	7542	8123.03	<b>7561.03</b>	7572.57	<u>13100.2</u>
st70	675	711.767	694.2	<b>682.067</b>	<u>1578.37</u>
eil76	538	570.767	559.1	<b>549.5</b>	<u>1155.9</u>
pr76	108159	119699	113510	<b>109395</b>	<u>252397</u>
kroA100	21282	22769.7	21747.8	<b>21352.5</b>	<u>72659.5</u>
rd100	7910	8415.43	8054.83	<b>7919.47</b>	<u>24980</u>
eil101	629	672.433	657.167	<b>633.3</b>	<u>1628.1</u>
lin105	14379	14888.2	14540.3	<b>14430.5</b>	<u>50828.4</u>
ch150	6528	6887.63	6676.47	<b>6578.67</b>	<u>26031.4</u>
rat195	2323	2524.67	2479	<b>2386.83</b>	<u>10503.4</u>
d198	15780	17235.3	16739.7	<b>16053.9</b>	<u>64996.9</u>
ts225	126643	135632	130436	<b>127427</b>	<u>793764</u>
a280	2579	2871.77	2847.17	<b>2704.5</b>	<u>16773</u>
lin318	42029	46927.4	46395	<b>43739.5</b>	<u>301153</u>
fl417	11861	13138.8	12741	<b>12303.9</b>	<u>188661</u>
pcb442	50778	58241.2	59163.1	<b>55502</b>	<u>418762</u>
rat575	6773	7877.9	7966.1	<b>7670.97</b>	<u>58253.8</u>
		<b>0 / 0</b>	<b>1 / 0</b>	<b>17 / 0</b>	<b>0 / 18</b>

La diferencia entre los resultados es clara, el algoritmo AGEDV necesita la diversificación voraz para no estancarse en óptimos locales. También es de notar la mejoría obtenida en el AG con el proceso de diversificación. Como se expuso en la sección III, el operador de mutación no es suficiente. Sin embargo, los resultados del AG+DV

están lejos de los del algoritmo AGEDV puesto que el resto componentes del algoritmo no aprovechan eficazmente la diversidad introducida.

TABLA VII  
COMPARACIÓN CON OTROS MECANISMOS DE PRESIÓN  
Y SELECCIÓN

Problema	Calidad Media		
	AGEDV	AGEDV sin Competición P.H Y con elitismo	AGEDV Con Selección Por Torneo
eil51	<b>427.267</b>	<u>1286.37</u>	439.467
berlin52	<b>7572.57</b>	<u>22970.9</u>	7902.8
st70	<b>682.067</b>	<u>2896.23</u>	697.2
eil76	<b>549.5</b>	<u>2134.83</u>	569.4
pr76	<b>109395</b>	<u>480546</u>	120715
kroA100	<b>21352.5</b>	<u>121662</u>	22749.8
rd100	<b>7919.47</b>	<u>47625</u>	8324.43
eil101	<b>633.3</b>	<u>2818.3</u>	648.233
lin105	<b>14430.5</b>	<u>95986.2</u>	14633.2
ch150	<b>6578.67</b>	<u>47638.1</u>	6940.2
rat195	<b>2386.83</b>	<u>19867.6</u>	2534.37
d198	<b>16053.9</b>	<u>163905</u>	16980.9
ts225	<b>127427</b>	<u>130959</u>	134465
a280	<b>2704.5</b>	<u>29621.5</u>	2852.07
lin318	<b>43739.5</b>	<u>518669</u>	48773
fl417	<b>12303.9</b>	<u>464044</u>	12852.8
pcb442	<b>55502</b>	<u>717566</u>	60669.3
rat575	<b>7670.97</b>	<u>95117</u>	8057.63
	<b>18 / 0</b>	<b>0 / 18</b>	<b>0 / 0</b>

La competición entre padres e hijos tiene vital importancia en el uso productivo de la diversidad. Permite seleccionar adecuadamente la mejor zona del espacio de soluciones. Si sustituimos dicho proceso por un modelo generacional elitista, la diversidad se dispara, obteniendo resultados nefastos como muestra la Tabla VII (columna 2). En esta tabla se incluyen además los resultados del algoritmo al sustituir la selección aleatoria adyacente por una selección por torneo binaria (columna 3). Como se ha mencionado en la sección III, esta última da lugar a una pérdida de diversidad, no aprovechando el potencial del algoritmo.

##### V. CONCLUSIONES Y TRABAJO FUTURO

En este trabajo se ha presentado un algoritmo genético con un mecanismo de diversificación que sustituye cromosomas iguales por soluciones generadas por algoritmos voraces aleatorizados, aplicando una competición entre padres e hijos y la selección aleatoria adyacente. El funcionamiento del algoritmo AGEDV pone en evidencia la importancia de la diversidad en la población de los AGs así como el aprovechamiento de la misma a través de los demás operadores para obtener el equilibrio entre exploración y explotación.

Como trabajo futuro se pretende avanzar en una doble perspectiva:

a) Analizar diferentes variantes del procedimiento diversificar y posibilidades para el criterio de

sustitución de soluciones dado por la función  $g(\cdot)$ , así como su aplicación en otros problemas.

b) Analizar la aplicación del procedimiento diversificar en otras metaheurísticas basadas en poblaciones.

##### REFERENCIAS

- [1] Forbes, N. (2004). *Imitation of life: how biology is inspiring computing*. Cambridge: Mit Press.
- [2] Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: The University of Michigan Press.
- [3] Goldberg, D. E. (2006). *Genetic algorithms*. Pearson Education India.
- [4] Dorigo, M., Maniezzo, V., & Colomi, A. (1996). *Ant system: optimization by a colony of cooperating agents*. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 26, no 1, p. 29-41.
- [5] McGinley, B., Maher, J., O'Riordan, C., & Morgan, F. (2011). *Maintaining healthy population diversity using adaptive crossover, mutation, and selection*. IEEE Transactions on Evolutionary Computation, vol. 15, no 5, p. 692-714.
- [6] Črepinsk, M., Liu, S. H., & Mernik, M. (2013). *Exploration and exploitation in evolutionary algorithms: a survey*. ACM Computing Surveys, vol. 45, no 3, p. 35.
- [7] Feo, T. A., & Resende, M. G. (1995). *Greedy randomized adaptive search procedures*. Journal of global optimization, vol. 6, no 2, p. 109-133.
- [8] Lozano, M., & García-Martínez, C. (2010). *Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report*. Computers & Operations Research, vol. 37, no 3, p. 481-497.
- [9] Rodriguez, F. J., Garcia-Martinez, C., & Lozano, M. (2012). *Hybrid metaheuristics based on evolutionary algorithms and simulated annealing: taxonomy, comparison, and synergy test*. IEEE Transactions on Evolutionary Computation, vol. 16, no 6, p. 787-800.
- [10] Storn, R., & Price, K. (1997). *Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces*. Journal of global optimization, vol. 11, no 4, p. 341-359.
- [11] Lenstra, J. K., Kan, A. R., & Shmoys, D. B. (1985). *The traveling salesman problem: a guided tour of combinatorial optimization*. New York: Wiley.
- [12] Rudolph, G. (1994). *Convergence analysis of canonical genetic algorithms*. IEEE Transactions on Neural Networks, vol. 5, no 1, p. 96-101.
- [13] Larrañaga, P., Kuijpers, C. M. H., Murga, R. H., Inza, I., & Dizdarevic, S. (1999). *Genetic algorithms for the travelling salesman problem: A review of representations and operators*. Artificial Intelligence Review, vol. 13, no 2, p. 129-170.
- [14] Reinelt, G. (1991). *TSPLIB—A traveling salesman problem library*. ORSA Journal on Computing, vol 3, no 4, p. 376-384. <http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/index.html>
- [15] Goldberg, D. E., & Deb, K. (1991). *A comparative analysis of selection schemes used in genetic algorithms*. G. J. E. Rawlins (Ed.), Foundations of Genetic Algorithms, p. 69-93
- [16] Eshelman, L. J. (1991). *The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination*. G. J. E. Rawlins (Ed.), Foundations of genetic algorithms, p. 265-283.
- [17] Krishnakumar, K. (1990). *Micro-genetic algorithms for stationary and non-stationary function optimization*. In 1989 Advances in Intelligent Robotics Systems Conference (p. 289-296). International Society for Optics and Photonics.
- [18] Simões, A., & Costa, E. (2011). *CHC-based algorithms for the dynamic traveling salesman problem*. C. Di Chio et al. (eds.), EvoApplications 2011, Part I, LNCS 6624, p. 354-363. Springer-Verlag.