Algoritmo Memético Equilibrado con Diversificación Voraz

Andrés Herrera-Poyatos[1] y Francisco Herrera[2]

[1] E.T.S. de Ingeniería Informática, Universidad de Granada
andreshp9@gmail.com
[2] Departamento de Ciencias de la Computación e Inteligencia Artificial, E.T.S. de Ing. Informática, Universidad de Granada
herrera@decsai.ugr.es

Resumen Los algoritmos meméticos, al igual que los algoritmos genéticos, pueden presentar una convergencia prematura cuando la diversidad de la población no es suficiente para mantener el equilibrio entre exploración y explotación de los operadores genéticos y la búsqueda local al generar nuevos cromosomas.
El algoritmo genético equilibrado con diversificación voraz [1] es una propuesta que intenta alcanzar el equilibrio entre exploración y explotación. En este trabajo extendemos este algoritmo al ámbito de los algoritmos meméticos. Analizamos la sinergia existente entre los operadores del algoritmo y la búsqueda local. Para ello se estudia el comportamiento en función el tamaño de la población, el número de soluciones generadas y el número de llamadas a la búsqueda local. Finalmente comparamos la propuesta con diferentes heurísticas del estado el arte basadas en el uso de la búsqueda local.

Palabras clave: algoritmos genéticos, algoritmos meméticos, diversidad versus convergencia, problema del viajante de comercio

1. Introducción

Los algoritmos meméticos [2] se basan en la hibridación de los algoritmos genéticos con una búsqueda local que mejore la calidad de los individuos de la población. Combinan los conceptos de la evolución natural y la genética con la evolución social y cultural representada por la búsqueda local. Se han convertido en una de las metaheurísticas más conocidas y utilizadas. Sin embargo, heredan los problemas de los algoritmos genéticos, siendo uno de estos la preservación de la diversidad de la población.

La diversidad de la población es sin duda una de las piedras angulares sobre las que gira el buen funcionamiento de los algoritmos genéticos y meméticos. 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. Además, en los algoritmos meméticos

se desperdiciará el tiempo de cómputo de la búsqueda local al iniciarse siempre la búsqueda en el mismo sector del espacio de soluciones. 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 [3], [4].

Una de estas propuestas es el denominado algoritmo genético equilibrado con diversificación voraz (AGDEV) [1], que aborda el problema de la diversidad mediante la diversificación voraz. Este mecanismo añade diversidad a la población en caso de que sea necesario sustituyendo aquellas soluciones que verifiquen determinado criterio de semejanza con otras de la población por nuevos cromosomas generados por un algoritmo voraz aleatorizado. Además, el algoritmo incluye diferentes componentes que ayudan a mantener la sinergia con la diversificación voraz.

En este trabajo extendemos el algoritmo AGDEV al ámbito de los algoritmos meméticos. A este nuevo modelo se le denomina algoritmo memético equilibrado con diversificación voraz (AMDEV). Además, analizamos la sinergia existente entre los operadores del algoritmo y la búsqueda local, utilizando como caso de estudio el conocido problema del viajante de comercio [5].

El trabajo se organiza en las siguientes secciones. En la Sección 2 se introducen los algoritmos meméticos y los operadores utilizados para el problema del viajante de comercio. En la Sección 3 se explica el funcionamiento del algoritmo AGEDV y se hibrida el mismo con una búsqueda local obteniendo el algoritmo AMEDV. En la Sección 4 se realiza un estudio experimental en el que se trata el tamaño de población y el comportamiento del modelo en comparación con el algoritmo AGEDV. Finalmente se contrastan los resultados con los de otras heurísticas del estado del arte basadas en el uso de la búsqueda local (algoritmo memético clásico, GRASP [6] e iterated greedy con búsqueda local [7]) y se analiza el número de llamadas a la búsqueda. En la Sección 5 se muestran las conclusiones obtenidas.

2. Algoritmos meméticos

En esta sección presentamos una breve introducción a los AMs. En la primera subsección se realiza una descripción de los mismos y se proporciona el pseudocódigo del algoritmo con el que se compara en la experimentación. En la segunda se estudia la aplicación de los AMs al problema del viajante de comercio.

2.1. Descripción de los algoritmos meméticos

Los algoritmos meméticos [2] son una heurística probabilística que combina los algoritmos genéticos y la busqueda local para conseguir un modelo con una mayor profundidad de búsqueda en el espacio de soluciones.

Un esquema habitual consiste en aplicar la búsqueda local una vez en cada iteración del algoritmo genético. El individuo de la población al que se aplica la búsqueda local es aquel que presenta un mejor valor de la función objetivo

pero no ha sido mejorado previamente por la búsqueda. De esta forma en cada iteración se aplica la búsqueda local a un elemento prometedor del espacio de soluciones, obteniendo un individuo de calidad.

La búsqueda local no solo permite mejorar la calidad de los cromosomas de la población sino que también introduce diversidad en la misma. Esto explica el mejor comportamiento de los algoritmos AMs frente a los AGs cuando la búsqueda local es de calidad.

El Algoritmo 1 muestra el algoritmo memético clásico utilizado en la experimentación. Consta de los operadores genéticos de selección, cruce y mutación y la aplicación de la búsqueda local descrita anteriormente. Por último utiliza un elitismo para mantener siempre la mejor solución encontrada en la población. Los parámetros α y β indican la probabilidad de cruce de una pareja y la probabilidad de mutación, respectivamente. Se establecen en 0,7 y 0,1, valores habituales en la literatura especializada.

Algoritmo 1 Algoritmo memético

1: Inicializar la población P con un algoritmo voraz aleatorizado.
2: **while not** CriterioParada **do**
3: Seleccionar $\frac{|P|}{2}$ parejas de individuos de P.
4: Cruzar cada pareja, con una probabilidad $\alpha \in (0, 1]$, obteniendo dos hijos por cada una.
5: Asignar a P' los nuevos hijos y las parejas que no se han cruzado.
6: Mutar cada solución con una probabilidad β.
7: Aplicar la búsqueda local al mejor individuo de P' no mejorado previamente (si existe).
8: Aplicar el elitismo: se sustituye la peor solución de P' por la mejor de P.
9: $P \leftarrow P'$
10: **fin while**

2.2. Aplicación al viajante de comercio

El estudio en este trabajo se realiza sobre el problema del viajante de comercio [5]. 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 objetivo 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, siendo además un problema utilizado con frecuencia en el estudio de los AMs, en el que constituyen una de las heurísticas más poderosas.

El problema consta con numerosos operadores genéticos. Como operador de selección se ha utilizado la selección por torneo, en la cual se toman k individuos de la población y se devuelve el mejor en términos de la función objetivo. El parámetro k se mantiene en $k = 2$. Además, implementamos el operador de cruce OX y la mutación por intercambio [8].

Una de las mejores heurísticas para el viajante de comercio es precisamente una búsqueda local conocida como Lin-Kernighan [9] y que utilizaremos como la búsqueda local de todos los algoritmos. Se ha optado por una versión moderna de la misma [10].

3. Algoritmo memético equilibrado con diversificación voraz

La sección actual se divide en dos subsecciones. En la primera subsección se introduce el algoritmo genético equilibrado con diversificación voraz mientras que en la segunda se extiende el algoritmo mediante la hibridación con una búsqueda local.

3.1. Algoritmo genético equilibrado con diversificación voraz

El algoritmo AGEDV se desarrolla a partir del problema de la diversidad en la población de los algoritmos genéticos [1]. La diversidad se formaliza como la media de la distancia entre todas las parejas de cromosomas. En el caso del viajante de comercio, la distancia entre dos soluciones, denotada $d(s, s')$, se puede definir como el número de arcos en los que difieren. Por tanto, la fórmula que proporciona la diversidad para una población P con n cromosomas es la siguiente:

$$D(P) = \frac{\sum_{s,s' \in P} d(s, s')}{n(n-1)}$$

La diversidad de la población en un algoritmo genético estándar se suele deber al operador de mutación. Sin embargo, el uso de este operador es a costa de empeorar de manera genérica la calidad de las soluciones. Además, la diversidad introducida es insuficiente como muestran los autores del algoritmo AGEDV en su estudio. Por ello, los autores proponen la diversificación voraz como mecanismo para aumentar la diversidad de la población sin dar lugar a una pérdida de calidad de las soluciones en el proceso. La diversificación voraz evita, además, una posible convergencia prematura del algoritmo.

La diversificación voraz consiste en la utilización de un algoritmo voraz aleatorizado para sustituir aquellos cromosomas que compartan determinadas características de similitud con otros de la población. Esto permite un aumento de la diversidad sin perder soluciones de calidad. Tras un estudio experimental los autores proponen sustituir aquellos cromosomas repetidos en la población.

El algoritmo AGEDV incluye la diversificación voraz junto a otras componentes que mejoran la sinergia entre todos los operadores. Sus características son:

1. Mecanismo de selección de padres que potencia la diversidad en el cruce llamado selección aleatoria adyacente.
2. Probabilidad de cruce igual a 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.

El Algoritmo 2 indica cómo se realiza una iteración del algoritmo AGEDV.

Algoritmo 2 Iteración del algoritmo AGEDV

Entrada: P : Población del algoritmo
1: Permutar de forma aleatoria los elementos de P.
2: Cruzar todas las parejas de cromosomas que se encuentren adyacentes en la nueva reordenación de P (también el primero con el último) obteniendo un hijo por pareja.
3: Cada hijo compite con su padre directo, sustituyéndolo en la población en caso de ser mejor en términos de la función objetivo.
4: Aplicar la diversificación voraz.

3.2. Algoritmo memético equilibrado con diversificación voraz

El algoritmo AMEDV se obtiene al extender el algoritmo AGEDV con una búsqueda local como la utilizada en el algoritmo memético de la Sección 2.1. Es preferible aplicar la búsqueda local tras la diversificación voraz para evitar que algún cruce haya introducido un individuo repetido en la población y este sea el que se mejore. El Algoritmo 3 muestra el pseudo-código del algoritmo AMEDV.

Algoritmo 3 Algoritmo memético equilibrado con diversificación voraz

1: Inicializar la población P con un algoritmo voraz aleatorizado.
2: **while not** CriterioParada **do**
3: Aplicar una iteración del algoritmo AGEDV a P (Algoritmo 2).
4: Aplicar la búsqueda local al mejor individuo de P' no mejorado previamente (si existe).
5: **fin while**

La Figura 1 muestra cómo evoluciona la diversidad de la población para los algoritmos AM y AMEDV durante una ejecución sobre la instancia ts225 [11]. El comportamiento es el observado en el estudio del algoritmo AGEDV [1]. La diversidad comienza en un valor alto debido a inicializar la población mediante un algoritmo voraz aleatorizado. Posteriormente, la diversidad disminuye conforme ambos algoritmos se centran en una zona del espacio de soluciones. Sin embargo, a partir de cierto punto, el algoritmo AMEDV mantiene la diversidad de la población en un valor medio gracias a la diversificación voraz y el funcionamiento del mismo, que permite trabajar con cromosomas de calidad en diferentes puntos del espacio de soluciones. Por el contrario, la diversidad sigue decreciendo en el algoritmo AM hasta que converge a un valor cercano a 0. El algoritmo ha convergido a un óptimo local y no es capaz de aumentar la diversidad sólo con el operador de mutación. Gracias a esta diferencia en el mantenimiento de la diversidad, el algoritmo AMEDV obtendrá mejores resultados, como se muestra en la Sección 4.

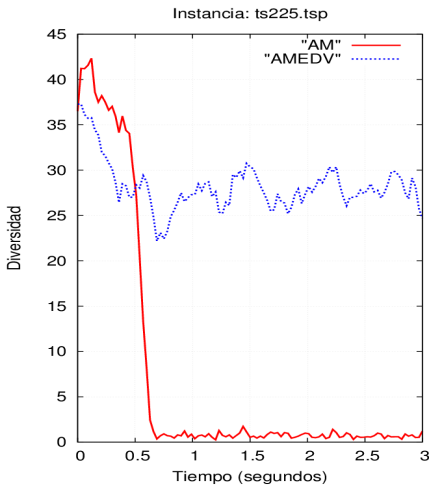

Figura 1: Diversidad en la población de los algoritmos AM y AMEDV.

4. 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 [11]. Todos los resultados consisten en la media de 30 ejecuciones del correspondiente algoritmo ejecutado durante $0{,}1N$ segundos, donde N es el número de ciudades que conforman la instancia.

La sección se divide en 3 subsecciones. En la primera se presenta un estudio sobre el tamaño de población para el algoritmo AMEDV. En la segunda se contrastan los resultados con los obtenidos para el algoritmo AGEDV en el que se basa la propuesta. En la tercera se compara el modelo con el algoritmo memético clásico y otras heurísticas basadas en búsqueda local del estado del arte desde una doble perspectiva: calidad de las soluciones y número de llamadas a la búsqueda local.

4.1. Análisis del tamaño de la población

La Tabla 1 proporciona los resultados obtenidos para el algoritmo AMEDV con los tamaños de población 8, 16, 32 y 64. Los resultados indican un mejor comportamiento del algoritmo cuando el tamaño de la población es pequeño. Esto se produce porque la mayoría del tiempo de cómputo es efectuado en la búsqueda local, realizándose pocas iteraciones de los operadores genéticos. Por tanto, se requiere de una mayor presión selectiva, proporcionada por un menor tamaño de población. Nótese que los algoritmos evolutivos usualmente necesitan un mayor tamaño población para evitar una convergencia prematura. Sin embargo, el algoritmo AMEDV carece de este problema gracias a la diversificación voraz.

Se han realizado estudios para un mayor tiempo de ejecución en los cuales el tamaño de población 8 es insuficiente para el buen comportamiento del algoritmo por la poca capacidad de exploración que proporciona. Sin embargo, una población de tamaño 16 sí es capaz de mantener la exploración necesaria a la vez que realiza una fuerte explotación, que no se consigue de forma tan eficaz para mayores tamaños de población. Por ello proponemos 16 como tamaño estándar, utilizándose este en el resto de la experimentación.

Tabla 1: AMEDV con diferentes tamaños de población para $0{,}1N$ segundos. Se destacan en negrita los modelos que obtienen el mejor resultado en una instancia y se subrayan los peores. La última fila muestra el número de veces que el modelo ha obtenido el mejor resultado frente al número de veces que ha obtenido el peor.

Problema	**Óptimo**	**Calidad Media**			
		TP = 8	TP = 16	TP = 32	TP = 64
eil51	426	**426**	426.167	426.867	426.933
berlin52	7542	**7542**	**7542**	**7542**	**7542**
st70	675	**675**	**675**	676	677.767
eil76	538	**538**	**538**	538.133	538.333
pr76	108159	**108159**	**108159**	**108159**	**108159**
kroA100	21282	**21282**	**21282**	**21282**	21282.8
rd100	7910	**7910**	**7910**	**7910**	7916.53
eil101	629	**629**	**629**	629.533	630.267
lin105	14379	**14379**	**14379**	**14379**	**14379**
ch150	6528	**6529.73**	6541.5	6547.3	6549.63
rat195	2323	**2326.67**	2329.4	2331.83	2334.47
d198	15780	**15794.7**	15801.4	15805.5	15815.3
ts225	126643	127036	126794	**126791**	126895
a280	2579	**2582.8**	**2582.8**	2590.8	2596.33
lin318	42029	42349.4	**42300**	42376.4	42444.9
fl417	11861	11948.8	**11940.8**	11948.7	11945.6
pcb442	50778	51438.2	**51257.1**	51438.3	51593.8
rat575	6773	6878.73	6874.23	**6869.27**	6878.77
		13 / 2	12 / 0	8 / 0	3 / 13

4.2. Comparación con el algoritmo genético equilibrado con diversificación voraz

La Tabla 2 compara la calidad media de las soluciones obtenidas por los algoritmos AMEDV y AGEDV con tamaños de población 16 y 64 respectivamente. El algoritmo AMEDV obtiene soluciones drásticamente mejores, mostrando la efectividad de la búsqueda local en combinación con los operadores genéticos y la diversificación voraz. De hecho, en muchas instancias casi siempre se alcanza el óptimo.

Se ha estudiado que el algoritmo AGEDV requiere un tamaño de población en torno a 60 para poder mantener una población diversa con individuos en diferentes puntos del espacio de soluciones [1]. Sin embargo, como se indica en la Sección 4.1, el algoritmo AMEDV necesita un menor tamaño de población. Se argumentaba este hecho por el menor número de iteraciones del algoritmo debido al coste computacional de la búsqueda local. Esto queda reflejado en la Tabla 2, que muestra el número medio de soluciones obtenidas en cada instancia durante la ejecución de ambos algoritmos. El algoritmo AGEDV genera entre 10 y 50 veces más soluciones que el algoritmo AMEDV. Sin embargo, la búsqueda local permite dar un salto de calidad a la población, siendo más efectivas las iteraciones del algoritmo AMEDV.

Tabla 2: AMEDV vs AGEDV con tamaño de población 16 y 64 respectivamente. Tiempo de ejecución $0{,}1N$ segundos. Se destacan en negrita los modelos que obtienen el mejor resultado en una instancia y se subrayan los peores.

Problema	**Óptimo**	**Calidad Media**		**Soluciones obtenidas**	
		AMEDV	AGEDV	AMEDV	AGEDV
eil51	426	**426.167**	427.267	128759	1692820
berlin52	7542	**7542**	7572.57	46387.8	1731320
st70	675	**675**	682.067	114622	1674870
eil76	538	**538**	549.5	127619	1740730
pr76	108159	**108159**	109395	60987.1	1377370
kroA100	21282	**21282**	21352.5	51970.3	14260
rd100	7910	**7910**	7919.47	54622.7	1473510
eil101	629	**629**	633.3	92876.7	1407060
lin105	14379	**14379**	14430.5	34887.5	538391
ch150	6528	**6541.5**	6578.67	50950	1270930
rat195	2323	**2329.4**	2386.83	47671	379744
d198	15780	**15801.4**	16053.9	16071.4	362111
ts225	126643	**126794**	127427	53980.6	724328
a280	2579	**2582.8**	2704.5	45372.8	612916
lin318	42029	**42300**	43739.5	10964.5	485157
fl417	11861	**11940.8**	12303.9	5499.87	422658
pcb442	50778	**51257.1**	55502	16309	231215
rat575	6773	**6874.23**	7670.97	3522.03	125132

4.3. Comparación con otras heurísticas basadas en búsqueda local

Las heurísticas basadas en búsqueda local se fundamentan en la aplicación de la búsqueda a elementos de zonas prometedoras del espacio de soluciones. Para que este proceso sea efectivo se requiere un mecanismo que proporcione soluciones de calidad a las que se aplique la búsqueda local. Algunas heurísticas, como GRASP e iterated greedy, utilizan para ello técnicas basadas en algoritmos

voraces. Otras, como los algoritmos meméticos, delegan en algoritmos evolutivos este trabajo. El algoritmo AMEDV combina lo mejor de ambos mundos pues la diversificación voraz garantiza que siempre se introduzcan nuevos cromosomas de calidad en la población y el carácter evolutivo del algoritmo genético permite que las soluciones sobre las que se aplica la búsqueda local sean cada vez mejores. Esta sinergia da lugar a los resultados de la Tabla 3. El algoritmo AMEDV obtiene holgadamente las mejores soluciones en todas las instancias. El algoritmo memético clásico es el segundo con mejores resultados gracias a que los elementos del espacio de soluciones sobre los que se aplica la búsqueda local mejoran con el tiempo. Nótese que el algoritmo AGEDV obtiene mejores resultados que el algoritmo memético clásico para instancias con menos de 110 ciudades (véase la Tabla 2) a pesar de que la mejora local utilizada, Lin-Kernighan, es una de las mejores heurísticas para el viajante de comercio.

Tabla 3: Calidad media de las soluciones obtenidas para los algoritmos AMEDV, AM, GRASP e IG. Tiempo de ejecución $0{,}1N$ segundos. Se destacan en negrita los modelos que obtienen el mejor resultado en una instancia y se subrayan los peores.

Problema	**Óptimo**	**Calidad Media**			
		AMEDV	AM	GRASP	IG
eil51	426	**426.167**	433.8	432.133	432.1
berlin52	7542	**7542**	7628.8	7670.43	7665.73
st70	675	**675**	689.567	692.433	692.133
eil76	538	**538**	548	551.867	550.733
pr76	108159	**108159**	109382	110083	110544
kroA100	21282	**21282**	21408	21469.2	21475.8
rd100	7910	**7910**	7951.87	8033.37	8061.1
eil101	629	**629**	641.533	647.833	648.267
lin105	14379	**14379**	14482.7	14551.6	14569.7
ch150	6528	**6541.5**	6575.73	6642.1	6671.73
rat195	2323	**2329.4**	2349.37	2373.23	2369.23
d198	15780	**15801.4**	15883.3	16015	15981.3
ts225	126643	**126794**	129022	129865	130035
a280	2579	**2582.8**	2637.43	2659.3	2660.7
lin318	42029	**42300**	42827.2	43068.8	43157.7
fl417	11861	**11940.8**	12041.1	12156.8	12130.5
pcb442	50778	**51257.1**	52319.9	52589.2	52585
rat575	6773	**6874.23**	6924.53	6944.6	6951.93

La primera columna de la Tabla 4 proporciona el número medio de llamadas a la búsqueda local para las ejecuciones de la Tabla 3. Podemos comprobar que el algoritmo memético clásico es el que más llamadas realiza. Esto se debe a que la búsqueda local tiene cada vez un menor tiempo de cómputo ya que la población

converge y, por ello, las soluciones sobre las que se aplica son cercanas a un óptimo local. Los algoritmos GRASP e IG son los que menos búsquedas locales realizan ya que en cada iteración la búsqueda local se aplica en una solución voraz o parcialmente voraz, respectivamente. Por tanto, el trabajo que tiene que hacer la búsqueda es elevado. Otra vez el algoritmo AMEDV vuelve a combinar lo mejor de ambos mundos, activando la búsqueda local sobre soluciones cada vez de mayor calidad que se cruzan con las soluciones voraces introducidas por la diversificación voraz, optimizando el tiempo de cómputo de la búsqueda local.

Tabla 4: Número de llamadas a la búsqueda local para los algoritmos AMEDV, AM, GRASP e IG. Tiempo de ejecución $0{,}1N$ segundos.

Problema	**Número de búsquedas locales**				**Porcentaje de iteraciones con búsqueda local**	
	AMEDV	AM	GRASP	IG	AMEDV	AM
eil51	7709.63	7478.47	3309.6	3256.13	100	100
berlin52	2759	4054.87	1863.37	1825.3	100	100
st70	6869.37	6690.73	2269.07	2247.83	100	100
eil76	7682.43	7756.63	2514.67	2479.27	100	100
pr76	3653.83	4940.3	722.633	712.6	100	100
kroA100	3096.7	4161.97	942.133	931.267	100	100
rd100	3268.77	4268.37	854.7	843.933	100	100
eil101	5543.07	5605.2	1390.47	1363.07	100	100
lin105	2085.3	3203.63	392.1	394.8	100	100
ch150	3067.7	3762.7	765.433	760	100	100
rat195	2871.6	4212.27	394.133	393.533	100	100
d198	968.467	1381.4	119.9	119.567	100	100
ts225	3284.07	4003.83	746.9	736.4	99.9918	100
a280	2729.97	3423.83	240.3	239.733	100	100
lin318	663.267	1093.1	56.4	56.0333	99.7588	100
fl417	331.533	586.8	33.0667	33.4333	100	100
pcb442	992.133	1765.13	74.1	74.9333	97.6112	100
rat575	216.3	1059.77	28.9667	27.8333	85.2672	100

El número de llamadas a la búsqueda local para los algoritmos GRASP e IG equivale al número de soluciones generadas. Sin embargo, en los algoritmos AM y AMEDV solo se aplica la búsqueda local en una iteración si existe una solución de la población que no haya sido mejorada previamente. En la segunda columna de la Tabla 4 se proporciona el porcentaje de iteraciones en las que sí se ha aplicado una búsqueda local a alguna solución de la población. En el caso del algoritmo memético podemos comprobar que este porcentaje es siempre 100 % pues casi toda la población es generada por el operador de cruce. Sin embargo, el algoritmo AMEDV solo aplica la búsqueda local si tras la competición entre padres e hijos se ha introducido una solución en la población. No obstante, el porcentaje sigue

siendo muy elevado, lo que prueba que el operador de cruce suele ser capaz de encontrar cromosomas mejores que alguno de sus padres, hecho que es clave para el buen comportamiento del algoritmo junto con la diversidad introducida por la diversificación voraz.

5. Conclusiones

En este trabajo se ha extendido el algoritmo genético con diversificación voraz al ámbito de los algoritmos meméticos. El funcionamiento del algoritmo se debe al proceso de diversificación voraz, ideado para solucionar el problema de la diversidad de los algoritmos genéticos. El buen comportamiento del modelo pone de manifiesto la necesidad de mantener la diversidad de la población en los algoritmos meméticos así como el aprovechamiento de la misma a través de los demás operadores para obtener un buen equilibrio entre exploración y explotación.

Se han contrastado los resultados del algoritmo con otras heurísticas basadas en búsqueda local del estado del arte, mostrando que para que la búsqueda local sea efectiva las soluciones sobre las que se aplique deben pertenecer a diferentes zonas del espacio de soluciones y tener una calidad razonable. Estos requisitos los verifica el algoritmo memético equilibrado con diversificación voraz y son los argumentos tras su buen funcionamiento.

Referencias

1. A. Herrera-Poyatos y F. Herrera, *Algoritmo genético equilibrado con diversificación voraz* Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados - MAEB, vol. 10, pp. 9-18, 2015.
2. Krasnogor, N., y Smith, J., *A tutorial for competent memetic algorithms: model, taxonomy, and design issues*, IEEE Transactions on Evolutionary Computation, vol. 9, no. 5, pp. 474-488, 2005.
3. Crepinsek, M., Liu, S. H. y Mernik, M., *Exploration and exploitation in evolutionary algorithms: a survey*, ACM Computing Surveys, vol. 45, no. 3, pp. 35, 2013.
4. McGinley, B., Maher, J., O'Riordan, C., y Morgan, F., *Maintaining healthy population diversity using adaptive crossover, mutation, and selection*, IEEE Transactions on Evolutionary Computation, vol. 15, no. 5, pp. 692-714, 2011.
5. Lenstra, J. K., Kan, A. R., y Shmoys, D. B. ,*The traveling salesman problem: a guided tour of combinatorial optimization*, New York: Wiley, 1985.
6. Feo, T. A., y Resende, M. G., *Greedy randomized adaptive search procedures*, Journal of global optimization, vol. 6, no. 2, pp. 109 - 133, 1995.
7. Ruiz, R. y Stützle, T., *A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem*, European Journal of Operational Research, vol. 177, no. 3, pp. 2033-2049, 2007.
8. Larrañaga, P., Kuijpers, C. M. H., Murga, R. H., Inza, I., y Dizdarevic, S., *Genetic algorithms for the travelling salesman problem: A review of representations and operators*, Artificial Intelligence Review, vol. 13, no. 2, pp. 129-170, 1999.
9. Lin, S. y Kernighan, B. W., *An effective heuristic algorithm for the traveling-salesman problem*, Operations research, vol. 21, no. 2, pp. 498-516, 1973.
10. Karapetyan, D., y Gutin, G., *Lin-Kernighan heuristic adaptations for the generalized traveling salesman problem*, European Journal of Operational Research, vol. 208, no 3, pp. 221-232, 2011.
11. Reinelt, G.,*TSPLIB - A traveling salesman problem library*, ORSA Journal on Computing, vol. 3, no. 4, pp. 376 - 384, 1991.
    `http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/index.html`