A Memetic Algorithm for the Traveling Salesman Problem

M. D. Arango and C. A. Serna

***Abstract*— This paper is designed a hybrid algorithm for solving the traveling salesman problem, which is listed among the NP-complete and which has received much attention in recent decades. As a solution strategy, we propose a memetic algorithm; which is based on the combination of an evolutionary algorithm and local search (2-opt), while the first analyzes several search spaces, the second explodes (improvement) each of the solutions found by the evolutionary algorithm. An individual analysis of the main heuristics used is made in the article.**

***Keywords*— traveling salesman problem, evolutionary algorithm, memetic algorithm.**

I. INTRODUCCIÓN

LOS PROBLEMAS de ruteo han sido ampliamente estudiados en las últimas décadas. Buscar maneras más eficientes de realizar la distribución de mercancías es una prioridad desde el ámbito logístico, reducir los costos es ampliar la competitividad, en la distribución logística son muchos los aspectos que pueden ser mejorados. En el transcurso de todos estos años se han formulado una amplia variedad de problemas de ruteo que tienen su estructuración más simple a partir del TSP el cual es un problema combinatorio fácil de formular que pertenece a la clase NP-completos, lo que significa que es difícil de resolver. El problema consiste en encontrar para un tour la menor ruta que visite un numero N de clientes en un ciclo Hamiltoniano, donde los clientes son puntos en un plano y las distancia entre cada uno de estos son tomadas en cuenta para la constitución del recorrido que se busca minimizar [1]. Debido al interés que ha recibido el TSP, en las últimas décadas se han formulado muchas heurísticas que tratan sobre su solución (para más detalle ver [2] y [3], entre estas se destacan las heurísticas evolutivas como los algoritmos genéticos (GA) los cuales han mostrado resultados eficientes en la solución de problemas combinatorios [4]. Algunos de estos trabajos se pueden ver en: [5], [6], [7], [8], y en [9] se puede encontrar una revisión de importantes trabajos, mientras que [10] realiza un análisis comparativo de diferentes aplicaciones de algoritmos genéticos al TSP. Sin embargo, y como señala [11] dos aspectos han despertado el interés de los últimos estudios en el campo: el primero relacionado con la codificación de una solución del problema en un cromosoma como lo exigen los GA, la dificultad de este campo reside en poder adaptar los problemas de ruteo que se derivan del TSP y que tienen varias restricciones a una codificación cromosomatica apropiada que represente o tenga en consideración las restricciones. El otro aspecto que ha despertado interés es el de mejorar el rendimiento de búsqueda global que desarrolla el GA y que es insuficiente para tratar problema combinatorios de la naturaleza NP- completos, dentro de estos estudios surgen precisamente los algoritmos meméticos (MA) los cuales combinan procesos de busque global refinados con heurísticas de búsqueda local.

El presente artículo se divide en las siguientes secciones: primero se hará una formulación básica del TSP sin mayores restricciones que las que supone pasar por cada nodo sólo una vez y regresar al origen, después se dará una definición de los MA y se hará referencia a algunos estudios en este campo. Posteriormente se presentarán la estructura del MA y los diferentes operadores heurísticos que lo componen. Al final se hace un análisis de resultados comparativo entre las diferentes técnicas usadas.

II. MODELO MATEMÁTICO PARA EL TSP

El TSP fue estudiado en el siglo XVIII por Hamilton y Kirkman y en el siglo XX Danzing amplió el concepto formulando problemas de varios vehículos, con ventanas de tiempo, capacitado etc. El TSP busca minimizar el costo total en el que se incurre al completar una ruta que pase por cada uno de los nodos establecidos. Sea V= $\{v_1, v_2, \ldots, v_n\}$ el conjunto de ciudades, clientes o nodos por visitar, A= $\{(i,j)$: i, $j \in V\}$ el conjunto de aristas y $d_{i,j}$ el costo asociado a la arista (i, j). Asumiendo que el vendedor tiene que visitar un número de ciudades V, iniciando y terminando en la ciudad de origen v_1, la representación de la ruta recorrida es llamada tour, el cual se constituye como un TSP. Su formulación es relativamente sencilla como se explica a continuación:

$$\text{min: } \sum_{i,j} c_{ij} x_{ij} \quad (1)$$

$$\sum_i x_{ij} = 1 \quad \forall j \in V \quad (2)$$

$$\sum_j x_{ij} = 1 \quad \forall i \in V \quad (3)$$

$$\sum_i \sum_j x_{ij} \geq 1 \quad i,j \in A(S,\bar{S}) \quad \forall S \subseteq V \setminus \{n\}, S \neq \emptyset, \quad (4)$$

$$con\ A(S,\bar{S}) = \{(i,j) \in A : i \in S, j /\in S\}$$

La función objetivo (1) se define a partir de la suma de los costos de recorrido entre cada vértice constituido por un origen i y un destino j. En cuanto a la primera restricción (2) asegura que cada destino j es visitado una vez y la restricción (3) establece que los vehículos abandonan cada nodo i solo una vez. La ecuación (4) usa los subconjuntos para eliminar las subrutas que pueden surgir y asegurar conectividad.

Si bien formular el modelo es relativamente fácil, su solución, al ser un modelo NP- completo, dista de ser sencilla cuando se usan métodos exactos (por ejemplo del tipo branch

M. D. Arango, Universidad Nacional de Colombia, Sede Medellín, Colombia, mdarango@unal.edu.co

C. A. Serna, Universidad San Buenaventura, Sede Medellín, Colombia, casernau@unal.edu.co

and bound) y el número de nodos se incrementa. Como propuesta de solución se han formulado heurísticas muy potentes entre las que se encuentran los MA, los cuales han entregado resultados muy satisfactorios. A continuación se describe su funcionamiento.

III. ALGORÍTMOS MEMÉTICOS

Por lo general en un GA las soluciones individuales no evolucionan durante su vida: se crean, son evaluadas, pueden ser seleccionadas como padres de nuevas soluciones y son destruidas. Sin embargo, la investigación sobre memética y algoritmos de búsqueda local genética ha demostrado que el rendimiento puede ser mejorado si las soluciones evolucionan durante su propia vida [12]. Los MA pertenecen a la clase de algoritmos evolutivos (AE) que aplican un proceso independiente de búsqueda local para refinar las soluciones. La combinación de la búsqueda global y local es una estrategia utilizada por muchas experiencias exitosas en optimización global. De hecho, el MA ha sido reconocido como un poderoso algoritmo de computación evolutiva [13]. Estos métodos están inspirados en los modelos de adaptación en los sistemas naturales que combinan la adaptación evolutiva de las poblaciones de las personas con problemas de aprendizaje individual dentro de toda la vida. Además, los MA se inspiran en el concepto de [14] de un meme, lo que representa una unidad de la evolución cultural que puede exhibir refinamiento local.

Desde el punto de vista de la optimización, los MA son AE híbridos que combinan la búsqueda global y local; los AE llevan a cabo la exploración, mientras que el método de búsqueda local explota la solución [15], [16]; es así como en diferentes contextos y situaciones, los MA también son conocidos como AE híbrido, buscadores genéticos locales, AE baldwinianos, AE lamarquianos y similares [17].

En la siguiente figura se muestra una comparación entre un GA y un MA a través de sus pseudocódigos.

Iniciar GA simple iniciar P; Para cada i ϵ P hacer evaluar (i); Mientras (condición de terminación no satisfecha) hacer P ← Seleccionar aleatoriamente dos padres i_a, i_b, ϵ P; i_c← Recombine (i_a, i_b) i_c ← mutar (i_c) Adicione i_c a P Fin (a)	Iniciar MA simple iniciar P; Para cada i ϵ P hacer i= Búsqueda local (i); Mientras (condición de terminación no satisfecha) hacer; P ← Seleccionar aleatoriamente dos padres i_a, i_b, ϵ P; i_c ← Recombinar (i_a, i_b); i_m ← mutar (i_c); i_m ← búsqueda local (i); adicione i_m a P; Si P converge entonces P ← Mutar & Búsqueda local (P); Fin (b)

Figura 1. (a) Pseudocódigo GA, (b) Pseudocódigo MA. Fuente: Elaboración propia

Entre los trabajos asociados a los MA como metodología de solución al TSP encontramos las siguientes publicaciones: En [18] implementan un algoritmo de búsqueda local 2-optmimal que usa un operador de recombinación llamado distance preserving recombination o DPX, mientras que el operador de mutación consiste en realizar tres cambios no secuenciales. [19] combinan varios algoritmos de búsqueda local como 2-opt, 3-opt y Lin–Kernighan los cuales son aplicados cada que un nuevo individuo es introducido a la población; en cuanto al operador de cruce usan el operador dropstar desarrollado por [20] y no usan operador de mutación. [1] usan un tipo de red neuronal llamada SOM (self-organizing map) incrustada en el AE, la dinámica evolutiva consiste en intercalar la ejecución SOM con un operador de asignación, la evaluación de la aptitud y un operador de selección, no se usa ningún operador de mutación. [21] realizar un conjunto de mutaciones de intercambio de genes *s* para *k* veces y realizar cruce uniforme parametrizado con la selección de torneo. Otros trabajos son [22], [23], [24], [25], [26], [27].

IV. ALGORITMO MEMÉTICO PROPUESTO.

Para el diseño del MA se partirá de un algoritmo genético el cual está compuesto por un operador de selección, uno de cruce, y otro de mutación, en cuanto a los algoritmos de búsqueda local se considerará la heurística 2-optimal y para el inicio de la población se usará un algoritmo GRASP (Greedy Randomized Adaptive Search Procedure). A continuación se explica el procedimiento que se usará en el MA y el GA.

A. Representación y función de evaluación

Con el fin de realizar una selección natural cada individuo *i* es evaluado en términos de su aptitud f_i, donde *f* es la función de evaluación la cual está directamente relacionada con la función objetivo (1) en las heurísticas tradicionales. El valor de aptitud (fitness value) mide la calidad de la solución y permite su comparación con otros individuos de la población dando lugar al proceso de selección explicado en la sección (4.3).

Otro de los aspectos importantes en un AE es la representación cromosomática que tendrá cada individuo. En el TSP los individuos corresponden a un posible recorrido el cual puede ser representado principalmente de las siguientes dos formas:

Representación basada en la trayectoria: es la forma más usada para el TSP. En ésta un recorrido se representa como un vector de n ciudades, en donde la *i*-esima posición la ocupa la *i*-esima ciudad que será visitada. De esta manera el recorrido *0-4-1-2-3-0* tiene la siguiente representación:

(4 1 2 3)

Representación basada en matriz binaria: En una representación binaria de un TSP con n ciudades, cada ciudad se codifica como una subristra de [log2 *n*] bits, donde [log2 *n*] denota la suma entre la parte entera del logaritmo en base 2 de *n* y la unidad. Una gira de n ciudades, se representará por medio de una ristra de [log2 *n*] bits. Si bien una de las formas más apropiadas para la representación cromosomática de un problema TSP, son los numero binarios estos traen cierta complicación cuando se aplican los operadores de cruce y más cuando se trata de un problema de ruteo, en donde los descendientes puede contener recorridos no validos por lo que se deben construir operadores de reparación (para mayor información ver [28]. Algunas otras representaciones binarias

se pueden encontrar en [29], [30], [31], [32]. En el GA que se presentará más adelante se usará la representación de [32] en la cual el elemento (*i*; *j*) de la matriz vale 1, si y solo si, en la gira la ciudad *j* se visita inmediatamente después de la ciudad *i*. La siguiente matriz (Fig. 2) representa la ruta *0 2 4 1 3 0*

	0	1	2	3	4
0	0	0	1	0	0
1	0	0	0	1	0
2	0	0	0	0	1
3	1	0	0	0	0
4	0	1	0	0	0

Figura 2. Representación binaria de un recorrido. Fuente: Elaboración propia

En nuestro MA usaremos la representación basada en trayectoria para los diferentes operadores evolutivos y de búsqueda local. Por otra parte la eficiencia del MA será comparada con los resultados de un GA que usará las dos representación: los operadores de evaluación, selección y mutación usaran la presentación basada en trayectoria y el operador de cruce la representación matricial binaria.

B. Inicio de población

Por lo general los AE inician con una población aleatoria de *N* individuos, en la que cada individuo representa una solución factible. Sin embargo el AE puede ser mejorado si en lugar de iniciar el proceso de evolución con individuos con bajos fitness, se concentra la búsqueda, desde el inicio, en una poblaciones con individuos de alto fitness, esto se logra con un algoritmo GRASP el cual es una metaheuristica multi arranque propuesta por [33], y que consiste básicamente en dos fases: una fase de construcción en la que tiene lugar la ejecución de una heurística Greedy aleatorizada y una segunda fase en la que se aplica una heurística de búsqueda local, en nuestro caso del tipo 2-opt.

Fase 1 Algoritmo de construcción Greedy aleatorizado:

Los algoritmos del tipo greedy o voraces son heurísticas intuitivas en las que la elección se realiza para maximizar o minimizar una función objetivo. Estas heurísticas son heurísticas de construcción en los que en cada paso de la ejecución se toma la decisión más favorable: para el TSP el algoritmo construye un tour $s \in S$ para *N* ciudades adicionando un nodo a la vez, y usando el criterio de la distancia más corta al último nodo adicionado. En la Fig. 3 se enseña un pseudocódigo para un algoritmo general y otro para un algoritmo donde el nodo siguiente se elige con una probabilidad p.

Fase 2. Procedimiento de búsqueda local 2-opt.

Uno de los operadores más importantes en el AM es el operador de búsqueda local en los cuales se concentra el mejoramiento de la descendencia en el GA. Este operador simula el desarrollo de individuos que puede darse en una generación *t* y que no están sujetos a las condiciones evolutivas, como cruce o mutación. Por lo general se usan heurísticas *k*-optimal, o metaheurísticas como búsqueda tabú, recocido simulado.

```
Iniciar Greedy (s ϵ S)
C:= {1, 2, 3, …, n};
Repetir
Encontrar k, 1 con g_kl = mini ϵ C
s_k:=l;
C:=C\{k};
Hasta C= Ø;
Retorne s;
Fin
```

(a)

```
Iniciar Greedy (s ϵ S)
C:= {1, 2, 3, …, n};
Repetir Hasta C= Ø;
Encontrar i donde g_k(1) = min f_ki, i ϵ C
Encontrar j donde g_k(2) = min f_kj, j ϵ C - {i}
Hacer p= g_k(0) / (g_k(0) + g_k(1))
Generar random [0,1]
Si random<p
s_k:= i;
Si no
s_k:= j;
Fin si
S=S U { s_k };
C=C - { s_k };
Retorne s;
Fin
```

(b)

Figura 3. (a) Pseudocódigo algoritmo Greedy. (b) Pseudocódigo algoritmo Greedy Aleatorizado. Fuente: Elaboración propia

En el presente análisis se usará la heurística 2-opt propuesto por [34] para iniciar la población y en el proceso de mejoramiento de la descendencia en el proceso evolutivo. El método 2-opt que consiste en realizar un movimiento que divide la ruta en dos subrutas y reconectarlas en una forma diferente (ver Fig. 4). Un pseudocódigo del algoritmo se explica a continuación:

Iniciar 2-Opt en x (x ϵ X) con k= número de nodos a ser intercambiados
g_{min}= Costo (x)
Para i=0 hasta k- 1
Para j= i+1 hasta k
x_{nueva} = 2-opt Reordenamiento (x, i, j)
g= Costo (x_{nueva})
Si g < g_{min}
x= x_{nueva}
fin Si
Fin

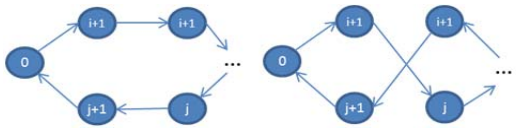

Figura 4. Representación gráfica algoritmo 2-opt. Fuente: Elaboración propia

C. Operador de selección

Entre los métodos de selección más usados se encuentran el método de la ruleta en el que a cada individuo se le asigna un valor indicando su calidad y la probabilidad de elegir este individuo es directamente proporcional a este valor. El inconveniente que surge con este operador es que la población muestra una rápida convergencia, reduciendo los espacios de búsqueda y haciendo más difícil encontrar soluciones cercanas a la solución óptima [35]. Por su parte el método de selección explicado en [36] la probabilidad de selección se distribuye de manera proporcional al rango del individuo lo que permite tener una mayor diversidad en las generaciones siguientes.

Otro método de selección muy extendido es el método de selección por torneo que combina características de los dos anteriores y en el cual se toma al azar un subgrupo de

individuos de la población, el mejor individuo en el subgrupo es seleccionado como padre y el procedimiento se repite hasta completar la población. Tanto en el GA como en el MA se usará este método de selección con un tamaño de subgrupo igual a 2.

D. Operador de Cruce

En el análisis se usarán dos operadores de cruce MX (matriz crossover) y DPX (distance preserving crossover). El MX, explicado en [28], es una extensión de los operadores de cruce basado en 1 punto y 2 puntos y que consiste en hacer uno o dos cortes al azar en los tours padres, los segmentos resultantes se combinan para dar origen a los descendientes (ver Fig. 5),

Este operador es usado en el GA, y usa una representación binaria de la solución y necesita un operador de reparación (MXr) que asegure descendencia factible. Por su parte, el DPX fue propuesto por [37] es un operador que sólo es eficiente con procedimientos de búsqueda local y consiste en generar descendencia cuya distancia a cada uno de los padres sea idéntica siguiendo el siguiente procedimiento: el contenido del primer padre es copiado en la descendencia y se borran todos los nodos que no pertenecen a segmentos comunes en el otro padre, posteriormente se realiza un procedimiento de reconexión aleatoria en el que se evita conexiones presentes en los padres (ver Fig. 6).

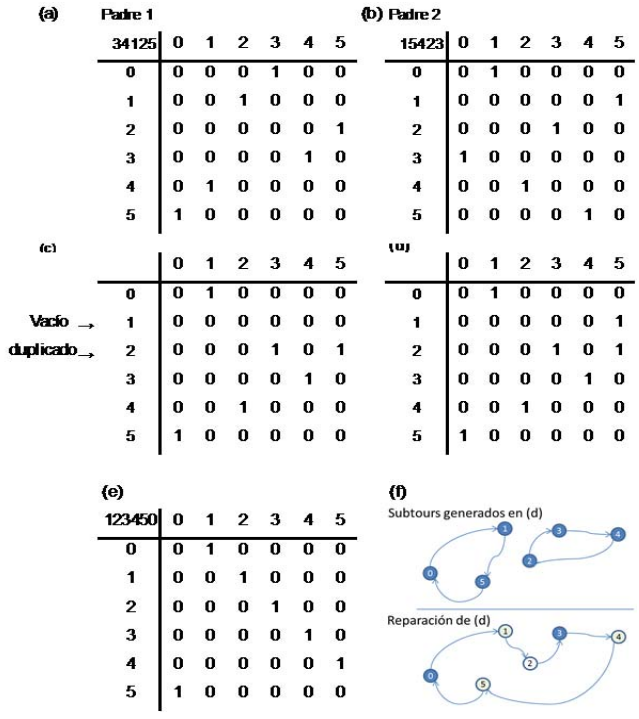

Figura 5. Operador MX. (a) y (b) Padres con punto de cruce en 1 y 3. (c) descendiente inviable, (d) Reparación I, (e) descendiente corregido reparación II. (f) Representación gráfica MXr. Fuente: Elaboración propia

En esta figura se enseña cómo obra el operador MX: a partir de dos padres (a) y (*b*) se genera un descendiente (*c*) el cual posee filas con 1´s duplicados y otras que carecen de éste. Para solucionar esta inconsistencia se aplica el operador de reparación (MXr) que genera un individuo (*d*) el cual puede llegar a generar subtours, como también se muestra en la figura; al ocurrir esto una segunda fase del operador de reparación se ejecuta para generar un tour valido (*e*) tratando de conservar vínculos (*i*, *j*) existentes en los padres

En la Fig. 6 se muestra como el operador guía el intercambio de información haciendo que la descendencia solo conserve los segmentos similares en los padres (a) y (*b*), los nodos restantes se incluyen siguiendo un algoritmo goloso similar a (*1*) procurando que los vértices (*i*, *j*) que se incluyan no estén incluidos en ninguno de los padres.

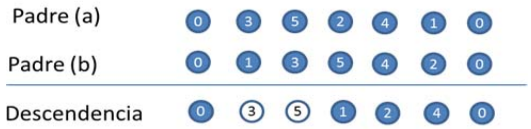

Figura 6. Operador DPX. Fuente: Elaboración propia

E. Operador de mutación

El operador de mutación basado en el cambio (SM) propuesto por [38], selecciona al azar un subtour en la solución para luego darles un nuevo orden de manera aleatoria. Otros operadores de mutación pueden consultarse en [39]. En la siguiente figura se elige el subtour *1 2 4* y se le aplica el operador SM

Figura 7. Operador SM .Fuente: Elaboración propia

A continuación se muestra el pseudocódigo del GA y el MA diseñados para resolver el TSP (Fig. 8):

Iniciar GA Iniciar P con Geedy Para cada i ϵ P hacer evaluar (i); Mientras (condición de terminación no satisfecha) hacer; P ← Seleccionar aleatoriamente dos padres ia, i_b, ϵ P; i_c← MX(i_a, i_b) i_{cr}← MXr (i_c) i_m ← SM (i_{cr}); Adicione i_m a P; Fin.	Iniciar MA Iniciar P con GRASP Para cada i ϵ P hacer evaluar (i); Mientras (condición de terminación no satisfecha) hacer; P ← Seleccionar aleatoriamente dos padres i_a, i_b, ϵ P; i_c← DPX(i_a, i_b) i_m ← SM (i_c); i_{mop} ← 2-Opt (im) Adicione i_{mo}p a P; Fin.
(a)	(b)

Figura 8. (a) Pseudocódigo GA. (b) Pseudocódigo MA. Fuente: Elaboración propia

V. RESULTADOS COMPUTACIONALES

En esta sección se presentan los resultados computacionales para evaluar la efectividad del MA propuesto y el GA canónico. El experimento se llevó a cabo a partir de una instancia para el TSP disponible en http://www.math.uwaterloo.ca/tsp/vlsi/pma343.tsp, la cual posee 343 nodos. Los algoritmos presentados en las secciones anteriores fueron ejecutados en un computador core i5, Ram 4 Gb.

Población: 200 individuos

Factor de mutación: 10%
Total corridas: 100.

TABLA I. RESULTADOS COMPARATIVOS.

	GRASP			A. GENÉTICO			A. MEMETICO	
N	Prom. Corridas	Mejor Sol.	% diferencia	Prom. Corridas	Mejor Sol.	% diferencia	Prom. Corridas	Mejor Sol.
50	184,4	184	0%	199	187	2%	184	184
100	383,8	381	1%	414	388	3%	377,8	378
200	797,6	793	2%	864	843	9%	780	776
343	1385,2	1374	3%	1590	1514	14%	1356	1328

El MA mostró tener un mejor desempeño para las cuatro instancias evaluadas (50, 100, 200 y 343 nodos). Así mismo el promedio en el MA está muy cerca de la mejor solución encontrada, lo que demuestra la forma en que obra esta metaheurística la cual explora exhaustivamente, a través del algoritmo de búsqueda local, los espacios de búsquedas que son establecidos desde el AE. En la Fig. 9 se muestra la mejor ruta encontrada para los 343 nodos.

Figura 9. Mejor ruta para 343 encontrada con un recorrido de 1328. Fuente: Elaboración propia

Para comprender los resultados en el GA, es necesario tener presente el carácter global de la búsqueda que realiza este tipo de algoritmo la cual analiza múltiples espacios de búsqueda. El GA mostró muy buenos resultados cuando se evaluaron 50 nodos en donde se obtuvo una solución con un costo 1% mayor a la mejor solución encontrada, pero cuando se incrementó el número de nodos a 100, 200 y 343 la diferencia con relación a la mejor solución también aumenta (2%, 3%, 9%, 14% respectivamente), esto es debido a que el GA necesita un mayor tamaño de población y de generaciones para ser más efectivo, lo que redunda en un mayor tiempo de procesamiento, de lo contrario el algoritmo puede converger rápidamente sin encontrar una solución de buena calidad.

Cuándo se inicia un AE con un algoritmo del tipo GRASP se corre el riesgo de limitar los espacios de búsqueda a una región de buenas soluciones pero alejada de la óptima, sin embargo también es válido mencionar que este algoritmo depura la población permitiendo explorar, a través del algoritmo evolutivo, soluciones potencialmente mejores; y es precisamente esta dinámica la que se refleja en la Fig. 10, donde las mejores soluciones para el algoritmo MA en cada corrida, tienen una alta convergencia, llegando a mostrar cierta estabilidad después de la décima corrida. En este punto, aunque al algoritmo se le hace más difícil encontrar mejores soluciones, amplía su espacio de búsqueda a partir de los operadores de mutación y cruce. Por otra parte, el GA no tiene una convergencia tan rápida y necesita de un mayor número de corridas para estabilizarse en una buena solución, según la Fig. 10 esto se logra en la corrida 81, sin embargo es posible que a partir de los operadores de cruce y mutación se encuentre una solución con mejor fitness.

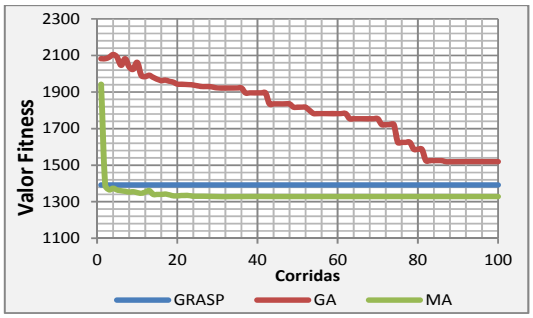

Figura 10. Solución MA con N= 343. Fuente: Elaboración propia

VI. CONCLUSIONES

En este trabajo se presentó un algoritmo memético para la solución del TSP, el cual integró metaheurísticas evolutivas y de búsqueda local. El MA fue comparado con un algoritmo GRASP y GA, mostrando un alto rendimiento en la solución encontrada. El algoritmo aquí presentado permite conocer de manera fácil cómo trabaja un MA que se inspira en el proceso de evolución natural y memética.

Si bien el GA, no mostró los mejores resultados de manera independiente, la integración de este con el algoritmo de búsqueda local es fundamental, dado que este último por si solo está limitado al número de espacios de búsqueda, mientras que el GA a través de los operadores evolutivos de selección, cruce y mutación los amplia permitiendo obtener mejores soluciones.

VII. REFERENCIAS

[1] J. Créput and A. Koukam, "A memetic neural network for the euclidean traveling salesman problem". *Neurocomputing*, v.72, p. 1250– 1264. Jan., 2009.

[2] R. Matai, S. P. Singh and M. L. Mittal, "Traveling Salesman Problem: An Overview of Applications, Formulations, and Solution Approaches". *Traveling Salesman Problem, Theory and Applications.* Prof. Donald Davendra (Ed.), December, 2010.

[3] D. Johnson and L. Mcgeoch, "Experimental Analysis of Heuristics for the ATSP". *The Traveling Salesman Problem and Its Variations.* Combinatorial Optimization. v.12, p. 369-443. 2007

[4] A. Tasan, and M. Gen, "A genetic algorithm based approach to vehicle routing problem with simultaneous pick-up and deliveries". *Computers & Industrial Engineering.* V.62, p. 755–761, Apr., 2012.

[5] C. Moon, J. Kim, G. Choi, and Y. Seo, "An efficient genetic algorithm for the traveling salesman problem with precedence constraints." *European Journal of Operational Research.* vl. 140, p. 606–617, 2002.

[6] H. Braun, "On Solving Travelling Salesman Problems by Genetic Algorithms". *Proc. First Workshop Parallel Problem Solving from Nature,* Springer Verlag, Berlin. v.496, p. 129-133, 1990.

[7] M. Karova, V. Smarkov, and S. Penev, "Genetic operators crossover and mutation in solving the TSP problem." *International Conference on Computer Systems and Technologies - CompSysTech*. 2005.

[8] F. Al-Dulaimi and H. A. Ali, "Enhanced Traveling Salesman Problem Solving by Genetic Algorithm Technique (TSPGA)." *World Academy of Science, Engineering and Technology*. v.38, p. 296-302. 2008.

[9] P. Larrañaga, C. Kuijpers, R. Murga, I. Inza and S. Dizdarevic, "Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators." *Artificial Intelligence Review*, V.13, P. 129–170, 1999.

[10] K. Bryant, "Genetic Algorithms and the Traveling Salesman Problem". Harvey Mudd College. December 2000.

[11] R. Cheng, M. Gen, and Y. Tsujimura, "A tutorial survey of job-shop scheduling problems using genetic algorithms, part II: hybrid genetic search strategies." *Computers & Industrial Engineering.* v. 36, No 2, p. 343-364. Apr. 1999.

[12] Marinakis, Y., y Marinaki M., "A hybrid genetic – Particle Swarm Optimization Algorithm for the vehicle routing problem." *Expert Systems with Applications.* v.37, No 2, pp. 1446–1455, Mar., 2010.

[13] R. Tavakkoli-Moghaddam, A. Saremi and M. Ziaee, "A memetic algorithm for a vehicle routing problem with backhauls." *Applied Mathematics and Computation.* v.181, No. 2, p.1049–1060. oct. 2006.

[14] R. Dawkins, *The Selfish Gene*, Oxford University Press, Oxford, UK., 1976.

[15] P. Moscato, C. Cotta, "A Modern Introduction to Memetic Algorithms", *Handbook of Metaheuristics,* 2nd edition, Michel Gendreau, Jean-Yves Potvin, Eds, International Series in Operations Research and Management Science. v.146, p. 141-183, 2010.

[16] P. Merz, *Memetic Algorithms for Combinatorial Optimization Problems: Fitness Landscapes and Effective Search Strategies*. PhD thesis, Department of Electrical Engineering and Computer Science, University of Siegen, Germany. 2000.

[17] A. El Fallahi, C. Prins, and R. Wolfler, "A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem." *Computers & Operations Research*; v.35, No. 5, p. 1725– 1741. May., 2008.

[18] K. Ghoseiri, H. Sarhadi. "A memetic algorithm for symmetric traveling salesman problem." *International Journal of Management Science and Engineering Management.* v.3, No 4, p. 275-283, feb. 2008.

[19] B. Bontoux, C. Artigues and D. Feillet, "A Memetic Algorithm with a large neighborhood crossover operator for the Generalized Traveling Salesman Problem." *Computers & Operations Research.*v.37, No. 11, p. 1844–1852. Nov. 2010.

[20] B. Bontoux, and D. Feillet, "Ant colony optimization for the traveling purchaser problem." *Computers & Operations Research*; v.35, pp. 628–37. 2008.

[21] F. Samanlioglu, W.G. Ferrell and M.E. Kurz, "A memetic random-key genetic algorithm for a symmetric multi-objective traveling salesman problem." *Computers & Industrial Engineering*. V.55, No 2, p. 439–449. Sep. 2008

[22] M. Rodrigues, and J. Ferreira, " Solving the Rural Postman Problem by Memetic Algorithms" *MIC'2001 - 4th Metaheuristics International Conference*, Porto, Portugal. 2001.

[23] Y. Wang and J. Qin, "A Memetic-Clustering-Based Evolution Strategy for Traveling Salesman Problems*". Lecture Notes in Computer Science*. v. 4481, p. 260-266. 2007.

[24] P. Földesi and J. Botzheim, "Modeling of loss aversion in solving fuzzy road transport traveling salesman problem using eugenic bacterial memetic algorithm." *Memetic Computing,* v.2, No 4, p. 259-271, Feb. 2010.

[25] L. Buriol, P. França and P. Moscato, "A New Memetic Algorithm for the Asymmetric Traveling Salesman Problem". *Journal of Heuristics*. V. 10, No. 5, p. 483-506, Sep., 2004.

[26] A. Piwońska and J.A Koszelew, "Memetic Algorithm for a Tour Planning in the Selective Travelling Salesman Problem on a Road Network". *Lecture Notes in Computer Science.* v.6804, p. 684-694. 2011.

[27] M. Shaikh and M. Panchal, "Solving Asymmetric Travelling Salesman Problem using Memetic Algorithm" *International Journal of Emerging Technology and Advanced Engineering.* v.2, No 11, p. 634- 639. Nov., 2012.

[28] A. Homaifar, S. Guan and G.E. Liepins, "Schema Analysis of the Traveling Salesman Problem Using Genetic Algorithms." *Complex Systems*; v.6, p. 533-552, 1992.

[29] M. Fox and M. McMahon, "Genetic operators for sequencing problems." *Foundations of Genetic Algorithms: First Workshop on the Foundations of Genetic Algorithms and Classifier Systems,* Rawlings, G. (ed.) Morgan Kaufmann Publishers, Los Altos, CA, p. 284-300. 1987.

[30] D. Whitley, T. Starkweather and D. Fuquay, "Scheduling problems and travelling salesman: The genetic edge recombination operator", *Proceedings on the Third International. Conference on Genetic Algorithms,* p. 133-140, 1989.

[31] D. Whitley, T. Starkweather and D. Shaner, "The traveling salesman and sequence scheduling: Quality solutions using genetic edge recombination", *Handbook of Genetic Algorithms*, Davis, L. (ed.) Van Nostrand Reinhold, p. 350-372. New York, 1991.

[32] D. Seniw, *A genetic algorithm for the traveling salesman problem*. M.Sc. Thesis, University of North Carolina at Charlotte. 1991.

[33] T.A. Feo, M. Resende, "Greedy randomized adaptive search procedures." *Journal of Global Optimization*, v.6, p. 109-133. 1995.

[34] G. Croes, "A method for solving traveling salesman problems" *Operations Res*, v.6, Nov., p. 791-81. 1958.

[35] A. Bjarnadóttir, *Solving the Vehicle Routing Problem with Genetic Algorithms*. Thesis MSc. Technical University of Denmark. 2004

[36] M. Melanie "An Introduction to Genetic Algorithms" A *Bradford Book,* The MIT Press Fifth printing. Massachusetts. 1999.

[37] B. Freisleben and P. Merz, "A Genetic Local Search Algorithm for Solving Symmetric and Asymmetric Traveling Salesman Problems." *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, IEEE Press, p. 616-621. May., 1996.

[38] G. Syswerda, "Schedule optimization using genetic algorithms". From: *Handbook of Genetic Algorithms*, L. Davis ,ed., Van Nostrand Reinhold, p. 332-349. New York 1991.

[39] S. Sivanandam and S. Deepa, *Introduction to Genetic Algorithms.* Springer ed. Berlin. 2008.

Martin Darío Arango es Ingeniero Industrial de la Universidad Autónoma Latinoamericana, especialista en docencia universitaria, Universidad Politécnica de Valencia, España. Magister en ingeniería de sistemas, Universidad Nacional de Colombia, Medellín, Doctor en Ingeniería Industrial, Politécnica de Valencia, España. Profesor Titular, Universidad Nacional de Colombia, Medellín, Colombia.

Conrado Augusto Serna Urán es Ingeniero Industrial de la Universidad Nacional de Colombia. Magister en Ingeniería administrativa, y candidato a Doctor en ingeniería industrial y organizaciones de la Universidad Nacional de Colombia, Medellín, Colombia. es Profesor Asociado de la Universidad de San Buenaventura, Medellín.