https://repository.uaeh.edu.mx/revistas/index.php/icbi/issue/archive
Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI

Pädi

ISSN: 2007-6363

Publicación Semestral Pädi Vol. 9 No. 17 (2021) 88-97

El problema del agente viajero resuelto mediante agrupación en clústeres y algoritmos genéticos

Traveling Salesman Problem solved by clustering and genetic algorithms

G. Anaya-Fuentes [a*]

[a] *Área Académica de Ingeniería y Arquitectura, Universidad Autónoma del Estado de Hidalgo, 42184, Pachuca, Hidalgo, México.*

Resumen

El presente artículo encuentra soluciones factibles para el Problema del Agente Viajero, mediante una nueva forma de agrupar al problema en clústeres con la intención de crear subproblemas del Agente Viajero, las cuales se resuelven por el metaheurístico algoritmos genéticos. Posteriormente las agrupaciones son unidas nuevamente utilizando las soluciones proporcionadas por el metaheurístico, obteniendo una solución final, además, la propuesta de agrupación de ciudades consiste en la utilización de la media aritmética sobre las coordenadas, para calcular iterativamente a los nodos representativos de cada familia. En la literatura se encuentra una tendencia para abordar este problema mediante la metodología propuesta. Los resultados demuestran que al utilizar esta metodología de agrupación se mejoran los resultados en comparación a las soluciones algoritmos genéticos sin utilizar clústeres.

Palabras Clave:
Agente viajero, algoritmos genéticos, agrupación, recorrido total, metaheurísticos.

Abstract

This article seeks to find feasible solutions for the Traveling Salesman Problem, by means of a new way of grouping the problem into clusters with the intention of creating Traveling Salesman subproblems, which are proposed to be solved by the metaheuristic genetic algorithm. Subsequently, the groupings are joined again using the solutions provided by the metaheuristic, obtaining a final solution. In addition, a way of grouping the cities of the problem is proposed using the arithmetic mean of the coordinates to iteratively calculate the representative nodes of each family. In the literature there is no similar method to solve the problem in question. The results show that using this grouping methodology improves the results compared to the genetic algorithms solutions without using clusters.

Keywords:
Traveling Salesman, genetic algorithms, clusters, fitness, metaheuristics.

1. Introducción

El Problema del Agente Viajero que en adelante denominamos como *PAV* es bastante conocido en la literatura, además de considerarse como uno de los problemas más difíciles de resolver, sin embargo, ha sido muy útil para resolver problemas de logística en la industria. El *PAV* consiste en encontrar el recorrido más corto al visitar *n* ciudades en una sola ocasión y regresando a la ciudad de origen. Este problema es clasificado como *NP-completo* y es considerado un gran reto para la ciencia. Los primeros que intentaron resolverlo fueron Dantzing, Fulkerson y Johnson, (1954), mediante una técnica denominada ramificación y acotamiento, utilizando una computadora IBM7090, en el cual se pudo percatar que el tiempo computacional para solucionarlo es demasiado alto. Recientemente este problema se ha abordado a través de Metaheurísticos como Colonia de hormigas (*ACO*), Recocido simulado (*RS*) y algoritmos genéticos (*AG*) por mencionar algunos.

El *PAV* ha sido abordado mediante diferentes técnicas, tal es el caso de Dahiya y Shabnam (2018), en el que identifica enfoques como el Método de ramificación y acotamiento, el algoritmo de Clarke y Wright, así como el método de Colonia de hormigas.

Además, se presentaron alternativas de solución al PAV mediante el heurístico denominado Bat (BA) en Panwar y Deep, (2021).

*Autor para la correspondencia: gustavoerick_anay@hotmail.com
Correo electrónico: gustavoerick_anay@hotmail.com (Gustavo Erick Anaya Fuentes)

Fecha de recepción: 31/03/2021 Fecha de aceptación: 04/05/2021 Fecha de publicación: 05/07/2021
https://doi.org/10.29057/icbi.v9i17.7130

También es posible resolver problemas adicionales mediante el PAV, tal es el caso del propuesto por Mosayebi, Sodhi y Wettergren, (2021), en el que se soluciona el problema de secuenciación de trabajos mediante el PAV, a su vez, es posible representar diferentes escenarios como robótica automática, mantenimiento de equipo, fabricación automatizada, con la intención de minimizar el tiempo de finalización del último trabajo.

La presente investigación propone agrupar a los nodos o ciudades del PAV en subproblemas denominados *clústeres*, en búsqueda de minimizar el recorrido en cada subproblema, utilizando para esto a los *AG*, posteriormente se propone un método de unión de *clústeres*, entregando recorridos factibles a bajo costo y en un tiempo computacional adecuado para la industria, por lo que a continuación analizamos la literatura de agrupación.

1.1. Antecedentes de agrupación en clústeres.

Dutta y Bhattacharya, (2015) propusieron técnicas de agrupación basadas en políticas y resolviendo a estas agrupaciones mediante algoritmos evolutivos; además, clasificaron a 7 tipos de agrupaciones que se basan en los siguientes: distancias, densidades, modelos, cuadros, semillas, espectros y jerarquías, con utilidad para minería de datos.

La agrupación en *clústeres* se ha utilizado para resolver diferentes problemas aplicados en distintos campos, por ejemplo, Nizam, (2010) propuso la agrupación como un poderoso sistema de control de la estabilidad de voltaje y presenta una nueva técnica de agrupación denominado Kohonen neural network. La formación de éstas puede simplificar el control del voltaje. Así también Vijayalakshmi, Jayanavithraa y Ramya, (2013) observaron en el campo de la genética en el que se miden miles de niveles de genes de manera simultánea, utilizando tecnología de microarreglos. En esta tecnología, el enfoque de *clústeres* mediante genéticos es utilizado para descubrir funciones similares entre genes. Bajo este enfoque se utilizan varios algoritmos de agrupación; como el propuesto por Vijayalakshmi *et al*. (2013) el cual es un algoritmo automático que provee de la habilidad de encontrar una fuerte convergencia global hacia una solución óptima. Weiya, Guohui y Dan (2015) investigaron un método novedoso llamado agrupación de gráfos de aproximación congruente, la solución obtenida mediante este método se aproxima al óptimo con una solución discreta. También se analizan las diferentes técnicas de agrupación para minería de datos por autores como Saroj and Chaudhary, (2015), debido a que agrupar es un tema de investigación activo en muchos campos como la estadística, en patrones de identificación y máquinas de aprendizaje. El análisis de la agrupación es una excelente herramienta para trabajar con una gran cantidad de datos. Por otra parte, Kaur y Kaur, (2012) utilizaron *clústeres* en Minería de datos mediante *K means* para dividir a los datos en *K* grupos; así también Nadana y Shriram, (2014) desarrollaron una metodología denominada *Megadata*, basada en un modelo de agrupación en *clústeres* para grandes conjuntos de datos. Los resultados experimentales demuestran que es posible encontrar una mejor calidad en la agrupación, pero sin mejorar el tiempo computacional.

Kaur y Singh, (2014) elaboraron un Algoritmo avanzado de agrupación en *clústeres* de manera que dirija a los grandes conjuntos de datos. Este método avanzado para agrupar permite calcular la distancia de cada objeto, además, requiere una simple estructura de datos para cada iteración. Sus resultados experimentales demostraron que el método avanzado del algoritmo de agrupación puede mejorar la efectividad de la velocidad y precisión del algoritmo, reduciendo la complejidad computacional. Tavse y Khandelwal, (2014) realizaron una clasificación de datos de internet en *clústeres*, para ser aplicados en la transmisión de datos, logrando una mejor eficiencia, mayor tiempo de vida de la red y estabilidad, optimizando la clasificación de los datos. Refianti, Mutiara, Juarna e Ikhsan, (2012) compararon 2 algoritmos denominados: propagación de afinidad y *k-means*, ambos agrupan datos referentes al tiempo de finalización de la tesis de licenciatura de estudiantes. Los resultados muestran que el algoritmo de propagación de afinidad proporciona resultados con más precisión en los datos de la familia y con más efectividad que *K-means*, mientras este último proporciona valores diferentes para los nodos representativos, después de cinco pruebas.

Nagy y Negru, (2014) analizaron métodos para agrupar en *clústeres* y que pueden ser usados para tratar patrones espaciales y temporales en una gran cantidad de datos. Estos autores utilizaron 55 nodos para aplicar los métodos de detección. Su enfoque permitió observar la existencia de diferentes familias espaciales y temporales. Por otra parte, Nidhi, (2015) propuso el Algoritmo *K-means* para el problema del incremento de datos, generado dinámicamente y sin repetición, el cual reduce el tiempo computacional, proporcionando resultados más precisos. Por lo que la agrupación inicial se realizó con los datos estadísticos, utilizando *K means*. Posteriormente para los próximos puntos, la mayor distancia entre el nodo representativo del *clúster* al más lejano de los puntos es utilizada para definir al próximo punto que está en la familia, repitiendo el proceso hasta cubrir el total de los datos.

Una de las heurísticas utilizada para resolver el *PAV* agrupado en *clústeres*, son los Algoritmos Genéticos (*AG)* en *clústeres* (*CAG)* presentados en el trabajo de Sivaraj y Ravichandran, (2011), quienes observaron que utilizando *CAG* se logra encontrar la solución óptima y en un tiempo menor al que se encuentra mediante los *AG* estándar *SGA*, esto se observó en tres diferentes casos mostrados en Sivaraj y Ravichandran, (2011). Este último autor desarrolló un mecanismo de aprendizaje no supervisado, usado para agrupar objetos similares en *clústeres*, asegurando que a pesar de las diferentes técnicas para agrupar a los clústeres que se encuentran disponibles, no hay una estrategia general que funcione igual en los diferentes problemas. Sin embargo, concluye que es mejor utilizar mecanismos simples al momento de formarlos. Liu, Dong, Lohse y Petrovic, (2016) propusieron un *GA*, intentando optimizar el consumo de energía, encontrándose con un problema multi-objetivo el cual mostró alta efectividad. Además, Li y Gao, (2016) utilizaron *GA* para resolver el Problema de secuenciación de trabajos mostrando buenos resultados en 201 instancias, lo cual, de manera análoga, permite resolver al *PAV*.

El PAV ha sido abordado mediante clústeres y algoritmos genéticos por Rani, Kholidah y Huda (2018), el algoritmo se utilizó para resolver un problema de itinerario para un turista, los resultados demuestran la factibilidad de este enfoque para resolver problemas de esta índole.

Por otra parte, Campuzano, Obregue y Aguayo (2020), presentan un enfoque algorítmico para el PAV asimétrico, mediante la generación eficiente de desigualdades válidas a partir de soluciones fraccionarias. Se utilizó el heurístico de Optimización

de colonia de hormigas para abordar este problema, el resultado del experimento muestra un mejor rendimiento del método propuesto que el método convencional de colonia de hormigas, en términos de velocidad de convergencia y mayor precisión de búsqueda.

El problema en cuestión se agrupa mediante clústeres en las investigaciones de Baniasadi, Foumani, Smith y Ejov, (2020), en donde la innovación principal se centra en la flexibilidad del algoritmo en comparación con otros de la misma naturaleza.

Barros, Jabba, Ardila, Guzman, y Ruiz (2021) utilizan una estrategia paralela para resolver el PAV mediante Algoritmos genéticos y Búsqueda Tabú.

Los métodos y técnicas mencionados anteriormente se han utilizado con la intención de agrupar a un conjunto de datos, los cuales forman parte de un problema a resolver, con la intención de reducir su complejidad computacional.

1.2. Antecedentes de agrupación en clústeres.

Para resolver el *PAV* agrupando en *clústeres*, se propone utilizar a los Algoritmos Genéticos (*AG)* como herramienta de búsqueda al problema presentado en cada grupo. Los *AG* son un Metaheurístico basado en la teoría de la evolución natural, los cuales cuentan con operadores genéticos tales como la selección, cruza y mutación de los individuos codificados para este problema según Vitayasak, Pongcharoen y Hicks, (2017). Los *AG* han sido utilizados para resolver el *PAV* y se han encontrado buenos resultados, sin embargo, hasta el día de hoy, ningún Metaheurístico asegura el valor óptimo; y las investigaciones buscan innovar sobre los mismos intentando hacerlos más eficientes con un menor tiempo computacional. Liu, Dong, Lohse , Petrovic, (2016) utilizaron *AG* para resolver un problema de optimización multi objetivo del consumo de energía y el desempeño del *Shop Floor Production*.

Los *AG* han sido utilizados para resolver problemas de asignación tales como los de Buffer (Hernández, Hernández, Jiménez, Hernández y Hernández, (2019).

Los AG siguen siendo un campo de investigación activo para el empleo de metaheurísticas en la optimización de sistemas cada vez más complejos, por lo cual sigue siendo valioso continuar con su desarrollo e investigación.

2. Metodología.

A continuación, se presenta la metodología propuesta en la presente investigación, en la cual se involucra la agrupación del *PAV* en *clústeres*, la búsqueda de su optimización mediante *AG* y posterior unión de *clústeres*, la cual brinda una solución final:

1. Identificar las coordenadas en X y Y del *PAV* a resolver.
2. Determinar el número de clústeres que agruparan a los nodos del *PAV* mediante la expresión $k = \sqrt{\frac{n}{2}}$ (Anaya, Hernández, Tuoh y Medina, 2018).
3. Generar aleatoriamente un nodo representativo de cada clúster, el cual es denominado como Centroide.
4. Asignar a cada nodo del *PAV* a un centroide, de acuerdo con su cercanía.
5. Iterativamente recalcular los centroides, tomando como nuevos centroides, al resultado de las medias aritméticas de las coordenadas en X y Y, de cada uno de los nodos asignados a cada centroide, tomando como criterio de paro, cuando los centroides no cambien de una iteración a otra.
6. Determinar los parámetros genéticos, tales como: Probabilidad de Cruza (*PC*), Probabilidad de Mutación (*PMu*), Número de individuos (*NumNind*) y el número de Iteraciones del AG (*Iteraciones*).
7. Generar población inicial.
8. Aplicar el operador genético Torneo en cada *clúster*.
9. Ejecutar el operador genético Cruza en cada *clúster*.
10. Realizar el operador genético mutación en cada *clúster*.
11. Repetir los pasos 8, 9 y 10, según el número de iteraciones, tomando a los resultados del paso 10 como población inicial en el paso 8.
12. Identificar la cercanía entre *clústeres*, para determinar la ruta de unión.
13. Unir los k clústeres y determinar la distancia de la ruta obtenida.

3. Resultados.

En la presente investigación se busca encontrar buenas soluciones factibles en un tiempo computacional razonable al PAV, utilizando la agrupación de las ciudades o nodos en clústeres, para resolver a cada uno de ellos utilizando AG. Para ello se propone la siguiente metodología, ejemplificando con un caso de la base de datos TSPLIB, (2020): La metodología utilizada es la Solución del PAV agrupado en clústeres, resolviendo mediante AG para ello se plantea el siguiente algoritmo:

1. Sea *n* el número de ciudades o nodos a visitar por el Agente Viajero. Como ejemplo tomamos las coordenadas en X y en Y de la instancia ulysses16.tsp del TSPLIB, (2020) y se muestran en la Tabla 1:

Tabla 1: Coordenadas de la instancia ulysses16.tsp del TSPLIB, (2020)

Nodo	Coordenadas en X	Coordenadas en Y
1	38.24	20.42
2	39.57	26.15
3	40.56	25.32
4	36.26	23.12
5	33.48	10.54
6	37.56	12.19
7	38.42	13.11
8	37.52	20.44

9	41.23	9.1
10	41.17	13.05
11	36.08	-5.21
12	38.47	15.13
13	38.15	15.35
14	37.51	15.17
15	35.49	14.32
16	39.36	19.56

2. Se calcula el número de grupos o *clústeres* en que se dividirá el problema y se le nombra por *k*. De tal manera que $k = \sqrt{\frac{n}{2}}$, redondeando el valor de *k* cuando es necesario. Para este ejemplo:

$$k = \sqrt{\frac{16}{2}} \approx 3$$

3. Los *k* clústeres son representados de forma individual, por nodos representativos de cada clúster, y se denominan centroides, los cuales se obtienen de coordenadas aleatorias en X y Y de la instancia del PAV objeto de estudio; para este caso los nodos representativos en X son *k* números aleatorios entre el valor mínimo de sus coordenadas 33.48 y el máximo de estas 41.23; de forma similar para Y el aleatorio se encuentra entre el mínimo -5.21 y el máximo 26.15.
4. Se asigna cada nodo a un *centroide* de acuerdo con la distancia más cercana entre nodo y *centroide*.
5. Con la intención de obtener centroides más representativos, estos se recalculan iterativamente mediante el valor esperado de las coordenadas en X y Y de cada *clúster*, es decir, en cada iteración se calcula la media aritmética de los nuevos valores en las coordenadas hasta que ya no cambien de una iteración a otra.

Para la instancia utilizada se obtiene el Centroide N= (Coordenada X, Coordenada Y), para *N= 1, 2,...,k* . En este caso se generan los siguientes nodos representativos: Centroide1 = (38.47, 15.13), Centroide2= (40.56, 25.32), Centroide3= (38.15, 15.35).

Posteriormente se agrupan los *n* nodos en *k* clústeres, asignando a cada uno de ellos al Centroide más cercano de tal manera que ningún nodo permanezca sin asignación de clúster. En la Tabla 2 se muestra la distancia entre cada nodo y cada Centroide, asignando cada nodo al Centroide del clúster más cercano, utilizando la expresión de la distancia entre dos puntos de Feliciano, A., Cuevas, V., Severino F., 2011:

$$d = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

Tabla 2: Asignación de clústeres mediante la distancia mínima del nodo al centroide.

Nodo	Centroide 1	Centroide 2	Centroide 3	Clúster asignado
1	5.2950	5.4215	5.0708	3
2	11.0748	1.2919	10.8930	2
3	10.4021	0.0000	10.2571	2
4	8.2900	4.8301	7.9966	2
5	6.7800	16.3883	6.7041	3
6	3.0776	13.4684	3.2146	1
7	2.0206	12.3961	2.2562	1
8	5.3943	5.7494	5.1288	3
9	6.6316	16.2338	6.9677	1
10	3.4083	12.2852	3.7961	1
11	20.4799	30.8569	20.6639	1
12	0.0000	10.4021	0.3883	1
13	0.3883	10.2571	0.0000	3
14	0.9608	10.5983	0.6648	3
15	3.0881	12.1122	2.8525	3
16	4.5185	5.8837	4.3804	3

Con ello se tienen asignados los siguientes nodos al Clúster 1= 6,7,9,10,11,12; Clúster 2= 2, 3, 4 y Clúster 3= 1, 5, 8, 13, 14, 15,16.

6. Ahora se aplica el AG en cada uno de los clústeres de manera individual, definiendo los parámetros de los operadores genéticos: Probabilidad de Cruza (*PC*), Probabilidad de Mutación (*PMu*), Número de individuos (*NumNind*) y el número de Iteraciones del AG (*Iteraciones*).

Para la instancia de este caso se asigna *PC = 0.9; PMu =0.8; NumInd = 3*n*, para este ejemplo *NumInd = 3*6 =18 y NumInd = 1*, en donde *n* para este caso es el tamaño del Clúster. Estos parámetros se obtienen basándose en la heurística de la investigación de (Anaya Fuentes, Hernández Gress, Seck Tuoh Mora, & Medina Marín, 2016).

7. De manera aleatoria, se genera una población inicial ϵ de *NI* individuos en los clústeres 1,2,3 … *k*, como se muestra en las Tablas 3, 4 y 5 respectivamente, para el ejemplo de ulysses16 del TSPLIB, (2020).

Tabla 3: Población inicial Ɛ del AG sobre el Clúster 1

Número de individuo	Individuo	Costo de recorrido
1	6-7-9-10-11-12	49.5424
2	6-9-11-7-10-12	44.6338
3	11-10-12-6-7-9	31.5979
4	9-6-10-7-12-11	33.7598
5	9-12-7-11-6-10	48.2949
6	10-6-7-9-11-12	45.5553
7	10-12-6-7-11-9	41.4226
8	6-9-11-7-12-10	43.9038
9	11-10-12-6-9-7	35.1362
10	9-6-10-11-12-7	49.9653
11	9-12-7-10-6-11	32.5767
12	10-6-7-12-11-9	42.6794
13	6-7-12-10-11-9	40.8529
14	6-9-12-11-7-10	53.1286
15	11-10-9-12-6-7	33.8752
16	9-6-11-10-7-12	45.9878
17	9-12-10-7-11-6	48.7222
18	10-6-12-7-9-11	28.9143

Tabla 4: Población inicial Ɛ del AG sobre el Clúster 2

Número de individuo	Individuo	Costo de recorrido
1	2-3-4	6.1220
2	3-2-4	5.7793
3	2-4-3	9.3175
4	3-4-2	9.3175
5	4-3-2	6.1220
6	4-2-3	5.7793
7	2-4-3	9.3175
8	3-2-4	5.7793
9	4-3-2	6.1220
10	3-2-4	5.7793

Tabla 5: Población inicial Ɛ del AG sobre el Clúster 3

Número de individuo	Individuo	Costo de recorrido
1	1-5-8-13-14-15-16	36.1588
2	16-14-15-13-8-5-1	36.5961
3	1-8-5-13-14-15-16	27.4875
4	14-16-15-13-8-5-1	40.9188
5	1-5-8-13-14-15-16	36.1588
6	15-16-14-8-13-1-5	37.7145
7	1-13-5-8-14-15-16	36.4432
8	16-13-14-15-8-5-1	35.3441
9	1-13-8-5-14-15-16	35.7361
10	14-13-16-15-8-5-1	39.6667
11	1-13-5-8-14-15-16	36.4432
12	15-8-16-14-13-1-5	29.9538
13	1-5-8-13-16-14-15	38.1241
14	16-14-15-1-13-8-5	34.5388
15	1-8-5-13-16-14-15	29.4528
16	14-16-15-1-13-8-5	38.8615
17	1-5-8-13-16-14-15	38.1241
18	15-16-14-5-8-13-1	38.3085
19	1-5-15-13-14-16-8	25.5688
20	16-14-1-15-13-8-5	35.4295
21	1-8-15-5-13-14-16	23.5821
22	16-14-1-15-13-8-5	35.4295

8. Posteriormente, se aplica el operador genético denominado Torneo del *AG*, para lo cual se generan *NI* parejas de números aleatorios discretos con rango de *1≤Número aleatorio≤NI.* El torneo consiste en elegir el individuo con menor costo entre cada pareja de *NI* individuos, como se muestra en las Tablas 6, 7 y 8; finalmente ordenándolos en forma ascendente como en las Tablas 9, 10 y 11.

Cada individuo ganador forma parte de la nueva población ϵ.

El clúster 1 tiene *n*=6, por lo que *NumInd=18.*

Para el Clúster 2, si *n*=3 el número de individuos es *NumInd=3*n=9*; donde *n* para este caso es el tamaño del *clúster*, sin embargo, al ser impar requiere agregar un individuo para realizar el torneo, por lo tanto *NumInd = 10* como se muestra en la Tabla 4.

Para el Clúster 3 el número de individuos es *NumInd=3*n=3*7=21*; donde *n* para este caso es el tamaño del Clúster, por ser impar *NumInd=22.* Como se observa en la Tabla 5.

Posteriormente compiten los pares de individuos, resultando vencedores aquellos que tienen el menor costo, como se muestran en las Tablas 6-8. En donde los individuos del Clúster1 que pasan al siguiente operador genético son: 2, 3, 6, 7, 9, 11, 13, 15 y 18. Mientras que el torneo para el Clúster2 tiene a los vencedores: 2, 3, 6, 8 y 10. Además de los individuos: 1, 3, 5, 8, 9, 12, 14, 15, 17, 19, 21 del Clúster3.

Tabla 6: Individuos a competir en Clúster 1

Individuo competidor	Individuo competidor	Mejor costo	Vencedor
1	2	44.6338	2
3	4	31.5979	3
5	6	45.5553	6
7	8	41.4226	7
9	10	35.1362	9
11	12	32.5767	11
13	14	40.8529	13
15	16	33.8752	15
17	18	28.9143	18

Posteriormente, los vencedores son ordenados de manera ascendente respecto a sus costos, como se muestra en las Tablas 9,10 y 11, para los clústeres 1,2 y 3 respectivamente.

Tabla 7: Individuos a competir en Clúster 2

Individuo competidor	Individuo competidor	Mejor costo	Vencedor
1	2	5.7793	2
3	4	9.3175	3
5	6	5.7793	6
7	8	5.7793	8
9	10	5.7793	10

Tabla 8: Individuos a competir en Clúster 3

Individuo competidor	Individuo competidor	Mejor costo	Vencedor
1	2	36.1588	1
3	4	27.4875	3
5	6	36.1588	5
7	8	35.3441	8
9	10	35.7361	9
11	12	29.9538	12
13	14	34.5388	14
15	16	29.4528	15
17	18	38.1241	17
19	20	25.5688	19
21	22	23.5821	21

Tabla 9: Individuos vencedores ordenados para el Clúster 1

Individuo vencedor	Costo
10-6-12-7-9-11	28.9143
11-10-12-6-7-9	31.5979
9-12-7-10-6-11	32.5767
11-10-9-12-6-7	33.8752

11-10-12-6-9-7	35.1362
6-7-12-10-11-9	40.8529
10-12-6-7-11-9	41.4226
6-9-11-7-10-12	44.6338
10-6-7-9-11-12	45.5553

Tabla 10: Individuos vencedores ordenados para el Clúster 2

Individuo vencedor	Costo
3-2-4	28.9143
4-2-3	31.5979
3-2-4	32.5767
3-2-4	33.8752
2-4-3	35.1362

Tabla 11: Individuos vencedores ordenados para el Clúster 3

Individuo vencedor	Costo
1-8-15-5-13-14-16	23.5821
1-5-15-13-14-16-8	25.5688
1-8-5-13-14-15-16	27.4875
1-8-5-13-16-14-15	29.4528
15-8-16-14-13-1-5	29.9538
16-14-15-1-13-8-5	34.5388
16-13-14-15-8-5-1	35.3441
1-13-8-5-14-15-16	35.7361
1-5-8-13-14-15-16	36.1588
1-5-8-13-14-15-16	36.1588
1-5-8-13-16-14-15	38.1241

9. Se compara la probabilidad del operador de cruza (*PC*), con un aleatorio denominado *Ap*, en donde $0 \leq Ap \leq 1$, de tal manera que la cruza se lleva a cabo si y solo si, $Ap \leq PC$. En otro caso se debe seguir directamente al Paso 10. Para el ejemplo de ulysses16 del TSPLIB, (2020), comparamos $PC=0.9$, con $Ap=0.8$, en cada par de Padres, de tal manera que se debe realizar la cruza.

La cruza se lleva a cabo a partir de dos padres, con la intención de generar a dos hijos; para lo cual se consideran dos puntos aleatorios de corte r_1 y r_2, sobre el Padre 1 y el Padre 2; en dónde $2 \leq r_1 \leq r_2 \leq n$. Los elementos del Padre1 que se encuentren entre r_1 y r_2, forman parte del Hijo 2, en las mismas posiciones de r_1 a r_2, el resto de los elementos del Hijo 2 se forman con aquellos nodos del Padre 2 que aún no están en el Hijo 2. De la misma forma los elementos del Padre 2 se encuentren entre r_1 y r_2 forman parte del Hijo 1, en las mismas posiciones de r_1 a r_2, el resto de los elementos del Hijo 1 se forman con aquellos nodos del Padre1 que aún no están en el Hijo 1. El resto de la población se somete al mismo procedimiento en caso de cumplir la condición $Ap \leq PC$.

Para este caso tomamos a los dos primeros padres *Clúster1*, *Padre1=10-6-12-7-9-11* y el *Padre2 =11-10-12-6-7-9*; considerando $r_1=2$ y $r_2=5$, como se muestra en la Figura 1.

El Segmento del *Padre 1 (12-7-9)* y el Segmento *Padre2 (12-6-7)* se cruzan, formando a los hijos 1 y 2, en este caso la cruza proporciona una solución no factible, puesto que no se cumple con un ciclo Hamiltoniano, con Hijo 1=10-6-12-6-7-11, Hijo 2 11-10-12-7-9-9, como se muestra en la Figura 2.

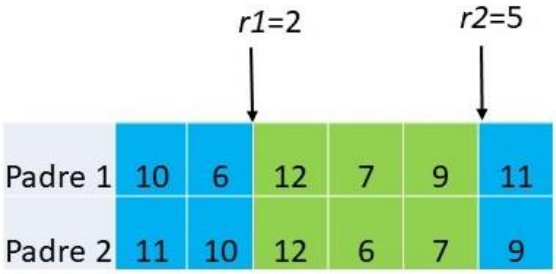

Figura 1. Padres 1 y 2 del Clúster 1, para cruza.

De acuerdo con Brito, D., Marin, L., y Ramírez, H., (2018). Un grafo es Hamiltoniano si tiene un ciclo que contenga todos los vértices del grafo, sin repetir ninguno.

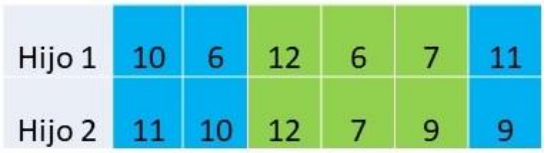

Figura 2. Hijos 1 y 2, después de la cruza, solución no factible.

Para hacer factible a las soluciones mostradas en la Figura 2, se identifica cuál de los nodos no se encuentra en la ruta, sustituyendo a esta por la sobrante, tal manera que los Hijos 1 y 2, quedan de la siguiente manera: Hijo 1= 10-6-12-9-7-11 e Hijo 2=11-10-12-7-6-9, con los costos 36.7857 y 30.4420, respectivamente, como se muestra en la Figura 3.

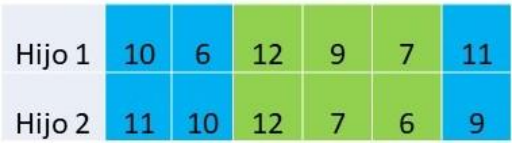

Figura 3. Hijos 1 y 2, después de la cruza, solución factible.

Los nuevos hijos se incluyen en la nueva población sustituyendo aquellos que tengan costos mayores.

Posteriormente se repiten los pasos 6 y 7 para el resto de la población; para esta instancia no se cumple la condición para cruzar en el resto de los individuos por lo que al ordenar los resultados en forma ascendente para los 3 Clústeres después del operador de cruza, tienen a la población que se muestra en la Tabla 12.

Tabla 12: Individuos después del operador de Cruza

Clúster 1	Clúster 2	Clúster 3
10-6-12-7-9-11	3-2-4	1-8-15-5-13-14-16
11-10-12-6-7-9	4-2-3	1-5-15-13-14-16-8
9-12-7-10-6-11	3-2-4	1-8-5-13-14-15-16
11-10-9-12-6-7	3-2-4	1-8-5-13-16-14-15
11-10-12-6-9-7	2-4-3	15-8-16-14-13-1-5
11-10-12-7-6-9		16-14-15-1-13-8-5
6-7-12-10-11-9		16-13-14-15-8-5-1
10-12-6-7-11-9		1-13-8-5-14-15-16
6-9-11-7-10-12		1-5-8-13-14-15-16

10. Posteriormente se realiza la mutación, con la siguiente condición: si $Apm \leq PMu$; donde *Apm* es un aleatorio en el intervalo $0 \leq Apm \leq 1$, en el caso contrario, no hay mutación

y acudimos al paso 11. Si la condición indica que se debe realizar la mutación, se eligen dos puntos de corte aleatorios rm_1 y rm_2, de manera que el individuo a mutar se divide en 3 elementos, a los que llamamos A, B, C. El individuo mutado se forma ordenando a los a los elementos como A-C-B. Para el ejemplo ulysses16 del TSPLIB, (2020), asumimos que se cumple la condición para mutar y tomamos al Hijo1 para mutarlo como muestra en la Tabla 13.

Tabla 23: Operador de Mutación

Individuo para mutar	Individuo mutado	Clúster
11-10-9-12-6-7	11-10-7-9-12-6	1
3-2-4	3-4-2	2
15-8-16-14-13-1-5	15-8-1-5-16-14-13	3

En este caso en particular las probabilidades de mutación permitieron mutar solamente a un individuo de cada *clúster*, por lo que el resto de la población sigue siendo similar.

Posteriormente ordenamos los resultados de forma ascendente en cada Clúster finalizando así la primera iteración del AG.

11. Se repiten los pasos 8, 9 y 10, tomando como población inicial la generación anterior, es decir, los resultados del paso 10, mostrados en la Tabla 14. Tomando como criterio de paro al número de iteraciones, el proceso se repite el número de *Iteraciones* previamente asignadas, por lo que para este ejemplo la población de cada Clúster después de aplicar *AG* en la siguiente iteración se muestra en la Tabla 14.

Tabla 14: Población de cada *Clúster* después de *AG*

Clúster 1	Clúster 2	Clúster 3
10-6-12-7-9-11	3-2-4	1-8-15-5-13-14-16
11-10-12-6-7-9	4-2-3	1-5-15-13-14-16-8
9-12-7-10-6-11	3-2-4	1-8-5-13-14-15-16
11-10-7-9-12-6	3-4-2	1-8-5-13-16-14-15
11-10-12-6-9-7	2-4-3	15-8-1-5-16-14-13
11-10-12-7-6-9		16-14-15-1-13-8-5
6-7-12-10-11-9		16-13-14-15-8-5-1
10-12-6-7-11-9		1-13-8-5-14-15-16
6-9-11-7-10-12		1-5-8-13-14-15-16

De tal manera que las mejores rutas factibles para cada *Clúster* son las siguientes: ruta *Clúster 1* =10-6-12-7-9-11, ruta *Clúster 2* =3-2-4 y ruta *Clúster 3*=1-8-15-5-13-14-16, con los costos: 28.9143, 5.7793 y 23.5821 respectivamente.

12. Una vez que se tienen *k* rutas con soluciones factibles para cada *clúster*, se realiza un procedimiento de unión de todos los clústeres para obtener una ruta final única; para ello, inicialmente se calcula la distancia del *Centroide 1* a cada uno del resto de los *Centroides*, con la intención de identificar cual es el *clúster* más cercano al *Clúster 1*; y este *clúster* es llamado *Clúster cercano (Cc)*. En el ejemplo de ulysses16 del TSPLIB, (2020), calculamos la distancia del *Centroide1* =(38.47, 15.13) a cada uno del resto de los *centroides*, utilizando la expresión de la distancia entre dos puntos $d=\sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$. Con lo que obtenemos la Tabla 15.

Tabla 35: Distancia del *Centroide1* a cada *Centroide*

Centroide	Centroide 2	Centroide 3
Centroide1	10.40	0.39

El *Centroide3* es el más cercano al *Centroide1*, como se observa en la Tabla 15, de tal manera que a *Centroide3* se le denomina *Clúster* cercano *Cc*.

13. Ahora se calcula la distancia entre *Centroide1* y cada uno de los nodos del *Cc*, eligiendo aquel nodo más cercano al *Centroide1*; y se le nombra como *nodoCc*, asimismo se calcula la distancia entre el centroide del *Cc* y cada uno de los nodos del *Clúster 1*, eligiendo aquel nodo más cercano a éste, al cual se le nombra como *nodoC1*. Posteriormente se une el *Clúster 1* con *Cc*, a través de los nodos *nodoC1* y *nodoCc*.

Para el ejemplo propuesto calculamos las distancias entre *Centroide1* y cada uno de los nodos *Cc*, en donde las Coordenadas del *Centroide1* son X = 38.47 y Y = 15.13, estas distancias se observan en la Tabla 16. En ésta, el nodo más cercano al *Centroide1* es el número 13, de tal manera que *nodoCc*=13. También calculamos la distancia del *clúster* Cc con coordenadas X = 38.15 y Y = 15.35 a cada nodo del *Clúster1*, como se observa en la Tabla 17.

De acuerdo a la Tabla 17 podemos encontrar al nodo del *Clúster 1* que es más cercano al *Centroide* de *Cc*, en este caso corresponde al nodo 12 por lo que *nodoC1=6*.

Tabla 46: Distancia de *Centroide* 1, a cada nodo de *Cc* (Clúster3)

Nodo	X	Y	Distancia a Centroide 1
1	38.24	20.42	5.2949
8	37.52	20.44	5.3943
15	35.49	14.32	3.0881
5	10.54	6.7799	6.7799
13	38.15	15.35	0.3883
14	37.51	15.17	0.9608
16	39.36	19.56	4.5185

Tabla 57: Distancia de Centroide del Cc a cada nodo de Clúster1

Nodo	X	Y	Distancia a Centroide Cc
10	41.17	13.05	3.7961
6	37.56	12.19	3.2146
12	38.47	15.13	0.3883
7	38.42	13.11	2.2562
9	41.23	9.1	6.9677
11	36.08	-5.21	20.6639

Se revisa cuál de los nodos unidos a *nodoCc*, se encuentra más cercano al mismo, con ello se elige el sentido del recorrido dentro del *clúster*. El último nodo de la ruta del *Cc*, es llamado *ClusterEndCc*, permaneciendo libre de acuerdo al algoritmo hasta unir al último *clúster* con él; de manera similar el último nodo del *clúster* 1 es llamado *ClusterEnd1*. Para el ejemplo en cuestión calculamos la distancia del nodo *nodoCc*=13,

respecto al nodo 5 y también al nodo 14, los cuales son contiguos. La Tabla 18 nos muestra tales distancias.

Tabla 68: Distancias del nodo 13 a sus nodos cercanos.

Nodo	Distancia al Nodo 13
5	6.7041
14	0.6648

Por lo que al ser el nodo 14 el más cercano, el sentido que debe seguir la ruta del *Cc* es 13-14-16-1-8-15-5; además *ClusterEndCc* = 5. Para encontrar el sentido del recorrido para el *Clúster 1*, es necesario encontrar el nodo más cercano al *nodoC1*=6, entre sus nodos contiguos, como se observa en la Tabla 19. De tal manera que al ser el nodo 12 es el más cercano al nodo 6, el sentido de la ruta del *Clúster1* es 6-12-7-9-11-10 además *ClusterEnd1*=10, como se muestra en la Figura 4.

Tabla 79: Distancias del nodo 6 a sus nodos cercanos

Nodo	Distancia al Nodo 6
10	3.7110
12	3.0776

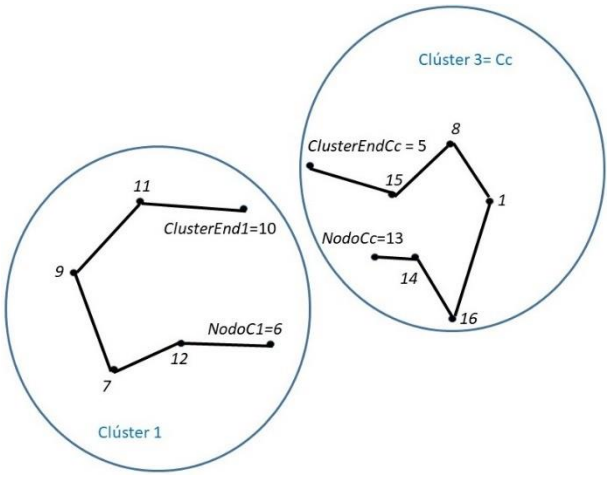

Figura 4. Sentido de recorrido de nodos en clústeres.

Los nodos *ClusterEnd1* y *ClusterEndCc* permanecen libres hasta que se unan al resto de los *clústeres*; en caso de que fueran los únicos *clústeres*, estos nodos se unen para obtener una ruta final. En caso de tener un mayor número de clústeres, es necesario seguir con el paso 13.

El siguiente clúster a unir se define encontrando la mínima distancia entre el *NodoCc* y cada uno de los *Centroides* restantes. Para este caso solo se tiene un *clúster* más por unir, de tal manera que se calcula la distancia de cada nodo de este *clúster* final correspondiente al *Clúster 2*, respecto al *NodoCc*, como se muestra en la Tabla 20.

Tabla 20: Distancias del *Nodo Cc* a los nodos de *clúster 2*

Nodo	Distancia respecto al *Nodo Cc (Nodo 13)*
3	10.2571
2	10.8929
4	7.9965

Con la Tabla anterior identificamos que el nodo del Clúster 2 más cercano al *NodoCc* es el 4; y nombramos a este como *NodoC2*. Posteriormente identificamos al nodo del cercano a *NodoC2* entre sus más cercanos del mismo *clúster* son: 1 y 10, para asignar el sentido de la ruta, para ello elaboramos la Tabla 21.

Tabla 21: Distancia entre nodos

Nodo	Distancia respecto al Nodo 4
3	4.8301
2	4.4874

Identificamos que el nodo más cercano al *NodoCc* es el 2, por lo que la secuencia del *Clúster2* es 4-2-3, además el último nodo es llamado *ClusterEnd2*, en este caso corresponde a 3.

14. El paso 13 se repite hasta *k* Clústeres.

15. Finalmente se une cada *nodoCn* con cada *ClusterEnd* subsecuente hasta *k clústeres*, para tener con ello la ruta final que cumple con el ciclo *Hamiltoniano* es 6-12-7-9-11-10-5-15-8-1-16-14-13-4-2-3-6 con un costo de recorrido de 97.7831, como se muestra en la Figura 5.

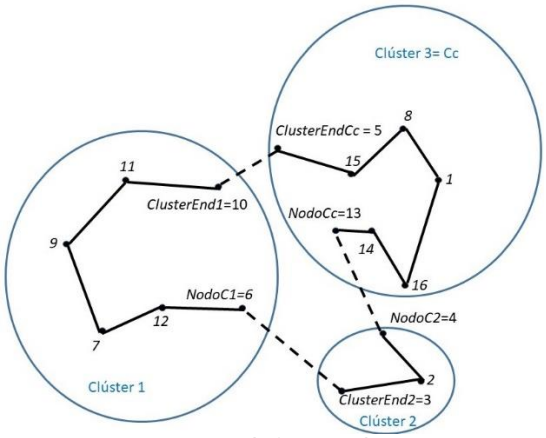

Figura 5. Unión final de clústeres

4. Comparación.

Se comparan tres métodos de *AG* de Hussain, Muhammad, Sajid, Hussain y Mohamd, (2017), denominadas PMX, OX, CX2, las cuales utilizan operadores genéticos para su solución, respecto al propuesto en el presente artículo. Los tamaños de las instancias son 21, 26, 33, 38, 42 y 53 nodos respectivamente, los cuales se observan en la Tabla 22.

Tabla 22: Comparación de la propuesta contra Hussain, Muhammad, Sajid, Hussain, y Mohamd, (2017)

Instancia	PMX	OX	CX2	CAG propuesto
gr21	2962	3005	2995	2313
fri26	1056	1051	1099	840
ftv33	1708	1804	1811	1786
ftv38	2345	2371	2252	1845
dantzing42	1298	1222	0699	655.07
ft53	13445	13826	10987	9745
kro124p	90231	97122	92450	46910
rbg443	6754	6932	6854	5995

Otra comparación se realiza en referencia al algoritmo propuesto en Kaabi y Harrath, (2019).

En la Tabla 22 es posible identificar beneficios del algoritmo implementado CAG, respecto a los operadores genéticos utilizados por Hussain, Muhammad, Sajid, Hussain, y Mohamd, (2017). Es necesario enfatizar que en cada uno de las instancias presentadas, se mejoraron los resultados.

Tabla 23: Comparación de la propuesta contra Kaabi y Harrath, (2019)

Instancia	(Kaabi & Harrath, 2019)	Propuesto
eil51	426	459
berlin52	7542	7154
st70	675	762
eil76	538	589
rat99	1211	1408
kroA100	21282	22684
ch130	6110	7146

Por otra parte, al comparar el algoritmo implementado CAG, contra el de Kabbi y Harrath, (2019), únicamente fue posible mejorar una de las distancias, por lo que los operadores genéticos utilizados, son mejores.

La Tabla 24 muestra la desviación estándar, media y mejor valor encontrado en las 30 corridas de cada una de las instancias reportadas.

Tabla 89: Distancias del nodo 6 a sus nodos cercanos

Instancia	Media	Desviación estándar	Mejor valor encontrado
gr21	3567	1241	2313
fri26	964	66	840
ftv33	2223	196	1786
ftv38	2256	200	1845
dantzing42	863	65	655
eil51	511	21	458
berlyn52	8806	559	7154
ft53	11202	994	9745
st70	821	39	761
eil76	647	23	589
rat99	1510	76	1407
kroA100	26668	1327	22684
kro124p	52224	2509	46910
ch130	7608	241	7146
rbg443	6870	707	5995

Se debe destacar que se encontraron mejoras en los resultados reportados en la literatura, en específico en la base de datos TSPLIB, en la que se reporta un valor de 7642 para la instancia Berlyn52, mientras que en esta investigación se determinó un valor de 7154 mediante la siguiente ruta: 17-3-18-31-22-1-49-32-45-19-41-8-9-10-43-33-51-11-52-13-14-47-26-27-28-12-25-4-6-15-5-48-24-38-37-40-39-36-35-34-44-46-16-29-50-20-23-30-21-42-7-2.

Los parámetros utilizados en por el algoritmo genético es posible destacar el número de individuos de cada instancia fue de 1000, el número de iteraciones 1000.

Los resultados obtenidos muestran que los resultados son satisfactorios pero que también falta trabajo por hacer para mejorar la efectividad del CAG para todas las instancias del PAV.

5. Conclusiones

En el presente artículo fue posible identificar los beneficios de los metaheurísticos como herramienta factible en la solución de problemas de asignación tales como: el Problema de asignación de trabajos, el problema del agente viajero, ruteo de vehículos, problema de asignación de Buffer, entre otros. Los cuales al considerarse problemas NP-Completos por su complejidad computacional tienen el problema de no alcanzar una solución óptima a partir de métodos de programación entera, por lo que el uso de metaheurísticos es la actual línea de investigación.

La propuesta del presente documento se centra en la combinación del uso de AG con la agrupación de elementos del PAV, mediante la innovación de una forma de agrupar sin generar una alta complejidad computacional, lo cual se comprueba derivado de las sencillas operaciones requeridas para su cálculo.

Futuras investigaciones podrían enfocarse en la optimización del número de agrupaciones en la búsqueda de minimización de recursos, también es posible utilizar otros metaheurísticos de manera individual o híbrida, tales como Clústeres con el vecino más cercano y algoritmos genéricos, Iterated Local Search, recocido simulado, entre otros, en la búsqueda de mejores resultados agrupando en clústeres.

Referencias

Anaya, G.E., Hernández, E.S., Seck, J.C., and Medina J., 2016. Solución al Problema de Secuenciación de Trabajos mediante el Problema del Agente Viajero, Revista Iberoamericana de Automática e Informática Industrial 13, 430–437. DOI:10.1016/j.riai.2016.07.003

Anaya, G.E., Hernández, E.S., Tuoh, J.C., Medina, J., 2018. Solution to travelling salesman problem by clusters and a modified multi-restart iterated local search metaheuristic. PLOS ONE, 1-20. DOI:10.1371/journal.pone.0201868

Baniasadi, P., Foumani, M., Smith-Miles, K., Ejov, V. 2020. A transformation technique for the clustered generalized traveling salesman problem with applications to logistics. European Journal of Operational Research, 285(2), 444-457. DOI: 10.1016/j.ejor.2020.01.053

Barros, A., Jabba, D., Ardila, C., Guzman, L., Ruiz, J. 2021. Adaptation of Parallel Framework to Solve Traveling Salesman Problem Using Genetic Algorithms and Tabu Search. Artificial Intelligence, 19.

Brito, D., Marin, L., y Ramírez, H., 2018. Ciclos Hamiltonianos que pasan a través de un bosque lineal en grafos bipartitos balanceados. Revista de matemática: Teoría y aplicaciones 25, 349-367. DOI: https://doi.org/10.15517/rmta.v25i2.33908

Campuzano, G., Obreque, C., Aguayo, M. 2020. Accelerating the Miller–Tucker–Zemlin model for the asymmetric traveling salesman problem. Expert Systems with Applications, 148, 113229. https://doi.org/10.1016/j.eswa.2020.113229

Dahiya, C., Sangwan, S., 2018. Literature Review on Traveling Salesman Problem. International Journal of Research 05, 1152-1155. DOI: https://journals.pen2print.org/index.php/ijr/article/view/15490/15018

Dantzig, G., Fulkerson, R., Johnson, S., 1954. Solution of a large-scale traveling salesman problem. Operations Research 2, 393-410.

Dutta, S., Bhattacharya, S.,A., 2015. Short review of clustering techniques. International Journal of Advanced Research in Management and Social Sciences, 131–139.

Feliciano, A., Cuevas, V., Severino F., 2011. Geometría analítica. Ed. UAI de la UAGro. ISBN 978-607-00-4156-3.

Hernández J., Hernández, S., Jiménez, J., Hernández, M., Hernández, I., 2017. Enfoque híbrido metaheurístico AG-RS para el problema de asignación del buffer que minimiza el inventario en proceso en líneas de producción abiertas en serie. Revista Iberoamericana de Automática e Informática Industrial 16, 447-458. DOI: 10.4995/riai.2017.10883

Hussain, A., Muhammad, Y., Sajid, M., Hussain, I., Shoukry, A., Gani, S., 2017. Genetic Algorithm for Traveling Salesman Problem with Modified

Cycle Crossover Operator. Computational Intelligence and Neuroscience, 1-7. DOI: 10.1155/2017/7430125

Kaabi, J., Harrath, Y., 2019. Permutation rules and genetic algorithm to solve the traveling salesman problem. Arab Journal of Basic and Applied Sciences 26, 283-291. DOI: 10.1080/25765299.2019.1615172

Kaur, G., Singh, D., Kaur, S., 2014. Pollination based optimitation for feature reduction at feature level fusion of speech and signature biometrics. IEEE, 1-6. DOI: 10.1109/ICRITO.2014.7014771

Kaur, N., Kaur J., 2012. Efficient k-means clustering algorithm using ranking method in data mining. International Journal of Advanced Research in Computer Engineering and Technology 1, 85–91.

Li, X., Gao, S., 2016. Cloud-Resolving Modeling of Convective. Springer, DOI**:**10.1007/978-3-319-26360-1

Liu, G., Song, S., Wu, C., 2012. Two Techniques to Improve the NEH Algorithm for Flow-Shop Scheduling Problems. Advanced Intelligent Computing Theories and Applications with Aspects of Artificial Intelligence of the series Lecture Notes in Computer Science 68, 41–48. DOI: 10.1007/978-3-642-25944-9_6

Liu, Y., Dong, H., Lohse, N., and Petrovic, S., 2016. A Multi-objective Genetic Algorithm for Optimisation of Energy Consumption and Shop Floor Production Performance. International Journal of Production Economics 179, 259-272. DOI:10.1016/j.ijpe.2016.06.019

Mosayebi, M., Sodhi, M., Wettergren, T. A. (2021). The Traveling Salesman Problem with Job-times (TSPJ). Computers & Operations Research, 129, 105226. DOI: https://doi.org/10.1016/j.cor.2021.105226.

Nadana, T., Shriram, R., 2014. Metadata based Clustering Model for Data Mining. Journal of Theoretical and Applied Information Technology, 59–64.

Nagy, M., Negru, D., 2014. Using clustering software for exploring spatial and temporal patterns in non-communicable diseases. European Scientific Journal 10, 37-47. DOI: 10.19044/esj.2014.v10n33p%25p

Nidhi, S. A., 2015. Modified Approach for Incremental k-Means Clustering Algorithm, 3, 1081–1084.

Nizam, M., 2010. Kohoen Neural Network Clustering for Voltage Control in Power Systems. Terakreditasi 51, 115-122.

Panwar, K., Deep, K. 2021. Discrete Grey Wolf Optimizer for symmetric travelling salesman problem. Applied Soft Computing, 105, 107298. DOI: 10.1016/j.asoc.2021.107298

Rani, S., Kholidah, K., Huda, S., 2018. A development of travel itinerary planning application using traveling salesman problem and k-means clustering approach. 7th International Conference on Software and Computer Applications, 327-331. DOI: https://doi.org/10.1145/3185089.3185142

Refianti, R., Mutiara, A., Juarna, A., Ikhsan, S., 2014. Analysis Implementation of algorithm clustering affnity propagation and k-means at data student based on gpa and duration of bachelor-thesis completion. Journal of Theoretical and Applied Information Technology 35, 69–76.

Saroj, Chaudhary, T., 2015. Study on Various Clustering Techniques. International Journal of Computer Science and Information Technologies 6, 3031-3033.

Sivaraj, R., Ravichandran, T., 2011. An Improved Clustering Based Genetic Algorithm for Solving Complex NP Problems. Journal of Computer Science 7, 1033-1037.

Tavse P., Khandelwal A., 2014. A critical Review on Data Clustering in Wireless Network. International Journal of Advanced Computer Research 4, 795–798.

TSPLIB, 2020. Base de datos Traveling Salesman Problem Library. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

Vijayalakshmi, S., Jayanavithraa, C., Ramya, L., 2013. Gene Expression Data Analysis Using Automatic Spectral MEQPSO Clustering Algorithm. International Journal of Advanced Research in Computer and Communication Engineering 2, 1145-1148.

Vitayasak, S., Pongcharoen, P., Hicks, C., 2017. A tool for solving stochastic facility layout problems with stochastic demand using either a Genetic Algorithm or modified Backtracking Search Algorithm. International Journal of Production Economics 190, 146-157. DOI: 10.1016/j.ijpe.2016.03.019.

Weiya, R., Guohui, L., Dan, T., 2015. Graph clustering by congruency approximation. The institution of Engineering and Technology, 841–849. DOI: 10.1049/iet-cvi.2014.0131