ECONSTOR
Make Your Publications Visible.

A Service of

ZBW Leibniz-Informationszentrum Wirtschaft
Leibniz Information Centre for Economics

Hamdan, Basma; Bashir, Hamdi; Cheaitou, Ali

Article

A novel clustering method for breaking down the symmetric multiple traveling salesman problem

Journal of Industrial Engineering and Management (JIEM)

Provided in Cooperation with:
The School of Industrial, Aerospace and Audiovisual Engineering of Terrassa (ESEIAAT), Universitat Politècnica de Catalunya (UPC)

Suggested Citation: Hamdan, Basma; Bashir, Hamdi; Cheaitou, Ali (2021) : A novel clustering method for breaking down the symmetric multiple traveling salesman problem, Journal of Industrial Engineering and Management (JIEM), ISSN 2013-0953, OmniaScience, Barcelona, Vol. 14, Iss. 2, pp. 199-218,
https://doi.org/10.3926/jiem.3287

This Version is available at:
https://hdl.handle.net/10419/261748

Standard-Nutzungsbedingungen:

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

Terms of use:

Documents in EconStor may be saved and copied for your personal and scholarly purposes.

You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.

If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.


https://creativecommons.org/licenses/by-nc/4.0/

WWW.ECONSTOR.EU

Mitglied der Leibniz-Gemeinschaft

OmniaScience

Journal of Industrial Engineering and Management
JIEM, 2021 – 14(2): 199-218 – Online ISSN: 2013-0953 – Print ISSN: 2013-8423
https://doi.org/10.3926/jiem.3287

A Novel Clustering Method for Breaking Down the Symmetric Multiple Traveling Salesman Problem

Basma Hamdan, Hamdi Bashir, Ali Cheaitou

University of Sharjah (United Arab Emirates)

eng_basmahamdan@yahoo.com, hbashir@sharjah.ac.ae, acheaitou@sharjah.ac.ae

Received: August 2020
Accepted: December 2020

Abstract:

Purpose: This study proposes a new two-stage clustering method to break down the symmetric multiple traveling salesman problem (mTSP) into several single standard traveling salesman problems, each of which can then be solved separately using a heuristic optimization algorithm.

Design/methodology/approach: In the initial stage, a modified form of factor analysis is used to identify clusters of cities. In the second stage, the cities are allocated to the identified clusters using an integer-programming model. A comparison with the k-means++ clustering algorithm, one of the most popular clustering algorithms, was made to evaluate the performance of the proposed method in terms of four objective criteria.

Findings: Computational results and comparison on 63 problems revealed that the proposed method is promising for producing quality clusters and thus for enhancing the performance of heuristic optimization algorithms in solving the mTSP.

Originality/value: Unlike previous studies, this study tackles the issue of improving the performance of clustering-based optimization approaches in solving the mTSP by proposing a new clustering method that produces better cluster solutions rather than by proposing a new or improved version of a heuristic optimization algorithm for finding optimal routes.

Keywords: clustering, factor analysis, k-means, multiple traveling salesmen

To cite this article:

Hamdan, B., Bashir, H., & Cheaitou, A. (2021). A novel clustering method for breaking down the symmetric multiple traveling salesman problem. *Journal of Industrial Engineering and Management*, 14(2), 199-218. https://doi.org/10.3926/jiem.3287

1. Introduction

1.1. The Context

The traveling salesman problem (TSP) is related to the determination of the shortest possible route connecting multiple cities so that a salesman can visit each city on his route only once and then return to his city of origin. A more complex version of the TSP is the multiple traveling salesman problem (*m*TSP), in which *m* salesmen must visit *n* cities, and each salesman must begin at and return to the same city. The *m*TSP problem is more challenging than the TSP because it requires identifying which cities must be allotted to each salesman as well as the corresponding optimal routes (Carter & Ragsdale, 2006).

Several variants of the *m*TSP exist, all of which are contingent on the number of depots involved (multiple or single); whether the paths are open or closed; whether the number of salesmen are set a priori, limited, or minimized; and whether additional constraints must be considered, such as a particular time frame for visiting each city, the maximum number of cities to be assigned to each salesman, the maximum and/or minimum allowable distance that a salesman must travel, and other issues (Bektas, 2006). The focus of this study is the single depot systematic *m*TSP. Throughout the rest of this paper, therefore, this variant is referred to as *m*TSP unless otherwise indicated.

The *m*TSP is of practical significance because it can be used to solve many everyday problems. Examples of the solved problems that have been reported in the literature include the printing-press scheduling problem (Carter & Ragsdale, 2006), the planning of autonomous mobile robots (Elango, Nachiappan & Tiwari, 2011; Trigui & Koub, 2016; Yu, Jinhai, Guochang, Rubo & Haiyan, 2002), unmanned aerial vehicles (Ann, Kim & Ahn, 2015; Lu, Zhang, He & Niu, 2016), production scheduling (Tang, Liu, Rong & Yang, 2000), designing networks for satellite systems (Boone, Sathyan & Cohen, 2015; Saleh & Chelouah, 2004), optimizing energy consumption in wireless sensor networks (Ma, Shi & Gu, 2018), routing vehicles (Expósito-izquierdo, Rossi & Sevaux, 2016) and drone deliveries (Kitjacharoenchai, Ventresca, Moshref-Javadi, Lee, Tanchoco & Brunese, 2019).

In the literature, several procedures have been developed for solving the *m*TSP, which is recognized as an NP-hard problem in combinatorial optimization (Bektas, 2006). One of type of these procedures is the utilization of a transformation procedure by which the *m*TSP is converted to a single standard TSP (Gorenstein, 1970). However, this type of conversion is not efficient because the resulting TSP can be highly degenerate, particularly if it involves a growing number of salesmen (Bektas, 2006; Chandran, Narendrananesh & Ganesh, 2006). A alternative transformation procedure that has proven to be more efficient is to break down the *m*TSP into several standard TSPs equal to the number of salesmen in the problem using a clustering algorithm; this allows each TSP to be solved individually using any heuristic optimization algorithm (Sofge, Schultz & De Jong, 2002). This two-stage procedure is referred to in this paper as a clustering-based optimization approach. A review of the studies that have proposed using this approach to solve the *m*TSP is presented in the following section.

1.2. Research Gaps and Justification for the Study

The performance of a clustering-based optimization approach to solve the *m*TSP depends on two components: (1) the performance of the clustering algorithm used to produce quality clusters; and (2) the performance of the heuristic optimization algorithm used to find the most optimal routes in the minimum computation time. Most researchers have paid less attention to the first component, as indicated by the fact that most clustering-based optimization approaches involve the use of the *k*-means clustering algorithm for forming clusters with a new or improved version of a heuristic optimization algorithm for finding optimal routes.

This study fills this gap and contributes to the literature by proposing a new clustering method as a means of improving the performance of clustering-based optimization approaches to solve the *m*TSP, with the following assumptions:

- All salesmen must begin and finish at one common single depot.
- Each salesman is assigned to a number of cities without considering any constraints.
- Each salesman must visit each of the assigned cities once per tour.
- Euclidean metric is used to measure the distance measure between every pair of cities.

The remainder of this paper is organized as follows. Section 2 is a brief review of most of the relevant literature. Section 3 details the proposed method. Section 4 presents the numerical study used to evaluate the performance of the proposed method. Finally, Section 5 concludes the paper and proposes opportunities for future study.

2. Literature Review

2.1. Heuristics Optimization Algorithms

Solving the *m*TSP is more difficult than solving the TSP because the former necessitates identifying which cities must be assigned to every salesman and determining the optimal route for every salesman. Because *m*TSPs are

NP-hard problems, heuristics optimization algorithms are most often used to solve them due their abilities to obtain near-optimal solutions within a reasonable CPU time (Bektas, 2006).

Russell (1977) proposed one of the earliest heuristic optimization algorithms by modifying an algorithm to solve the *m*TSP that Lin and Kernighan (1973) developed to solve the TSP. Since then, many heuristics optimization algorithms have been advanced in the literature, including genetic algorithms (e.g., Carter & Ragsdale, 2006; Király & Abonyi, 2010; Larki & Yousefikhoshbakht, 2014; Lu et al., 2016; Sedighpour, Yousefikhoshbakht & Darani, 2011; Singh & Baghel, 2009; Snyder & Daskin, 2006; Tang et al., 2000; Yuan, Skinner, Huang & Liu, 2013), neural networks (e.g., Modares, Somhom & Enkawa, 1999; Somhom, Modares & Enkawa,, 1999), simulated annealing (e.g., Ann et al., 2015; Paydar, Mahdavi, Sharafuddin & Solimanpur, 2010), and ant colony algorithms (e.g., Ghafurian & Javadian, 2011; Liu, Li & Zhao, 2009). Harrath, Salman, Alqaddoumi, Hasan and Radhi (2019) recent study combined a modified ant colony, the 2-opt, and a genetic algorithm to solve *m*TSPs.

2.2. Clustering-Based Optimization Approaches

Many studies have proposed a two-stage solution procedure to improve the performance of heuristics optimization algorithms to solve large *m*TSPs. The first stage uses a clustering method to group the cities into several clusters, whereas the second stage involves finding a Hamiltonian route into each cluster using a heuristic optimization algorithm. One of the earliest studies to propose solving the *m*TSP using a clustering-based optimization approaches was that of Sofge et al. (2002): They used the neighborhood attractor schema combined with different heuristic optimization algorithms, including a shrink-wrap algorithm and an assortment of evolutionary computation algorithms. The resulting combinations were tested on three different problems, and the encouraging results opened a new avenue of research in solving the *m*TSP (Sofge et al., 2002). Since then, several studies have developed clustering-based optimization approaches that combine a clustering algorithm with one or more heuristic optimization algorithms. These studies are summarized in Table 1.

Study	Clustering Algorithms used to Break Down the *m*TSP				Optimization Algorithms used to Find the Optimal Routes					
	Genetic algorithm	Standard k-means	Neighborhood attractor schema	Ant colony	Genetic algorithm	Generational Monte Carlo optimization	Nearest neighbor algorithm	Particle swarm optimization	Simulated annealing	Tabu search
Sofge et al. (2002)			✓		✓	✓		✓		
Sze & Tiong (2007)		✓			✓		✓			
Nallusamy, Duraiswamy, Dhanalaksmi and Parthiban (2009b)		✓			✓					
Nallusamy, Duraiswamy, Dhanalaksmi and Parthiban (2009a)		✓							✓	✓
Sadiq (2012)		✓			✓					
Necula, Breaban and Raschip (2015)		✓		✓						
Lu et al. (2016)		✓			✓					
Mardiyati, Safitri and Jihan., (2017)		✓		✓						
Shabanpour, Yadollahi and Hasani (2017)	✓				✓					
Zeebaree, Haron, Abdulazeez and Zeebaree (2017)		✓			✓					
Kovács, Agárdi and Debreceni (2018)		✓			✓					

Table 1. The clustering and optimization algorithms used in various studies

Two general observations can be drawn from Table 1. First, to break down the *m*TSP, 91 percent (10 out of 11) of the studies used either the classical version of the *k*-means or one of its variants; 10 studies used the standard version of the *k*-means; and only one study (Sofge et al., 2002) used a neighborhood attractor schema, a variant of the *k*-means. Second, the majority of studies (72.7%) used genetic algorithms to find the optimal routes.

In addition to the studies presented in Table 1, Chandran et al. (2006) and Ma et al. (2018) developed their own clustering algorithms to break down the *m*TSP. However, the performance of Chandran et al.'s (2006) algorithm was not evaluated with respect to the extent to which it improves the performance of heuristics optimization algorithms in solving the *m*TSP; it was only evaluated with respect to producing quality clusters using only one measure (relative percentage deviation). Ma et al. (2018) evaluated the performance of the developed clustering algorithm only in terms of total travel distance. This evaluation was achieved by combining a clustering algorithm with integer linear programming to determine the optimal routes.

2.3. The *k*-Mean Algorithm

The standard version of this algorithm, Lloyd's algorithm, is named for Stuart Lloyd, who first proposed it in 1957 (Morissette & Chartier, 2013). Because of its efficiency in grouping large data sets into good quality clusters, the k-means is by far the most extensively used clustering algorithm (Kovács et al., 2018; Morissette & Chartier, 2013; Singh, Malik & Sharma, 2012).

In the *k*-means algorithm, a large number of data points are grouped in a number of clusters to minimize the intra-cluster variance. The first step of this algorithm is to randomly choose the locations of the initial centroid for the required number of clusters. Each data point is then apportioned to its nearest centroid, which is subsequently changed based on the data points that are allotted to the cluster. This process is continued until the values of centroids become stabilized.

Literature has highlighted several limitations of the *k*-means algorithm (see Kaur & Kaur, 2013; Singh et al., 2012). These limitations include its sensitivities to the presence of outliers, the formation of unbalanced or even empty clusters, and its performance depends heavily on initialization (selection of initial centroids), so poor initialization can lead to poor performance (Fränti & Sieranoja, 2019). The last limitation can be surmounted by rerunning the algorithm many times using different random locations for the starting centroids in order to identify the best clustering solution or by using a better initialization technique. In this regard, several variants of the standard version of the *k*-means have been proposed in the literature, including Forgy's algorithm (Forgy, 1965), MacQueen's algorithm (MacQueen, 1967), the neighbourhood attractor schema (Sofge et al., 2002), *k*-means ++ algorithm (Arthur & Vassilvitskii, 2007), and the seeding algorithm for *k*-means problem with penalties (Li, Xu, Yue, Zhang & Zhang, 2020; Li, Xu, Zhang & Zhou, 2020). These variants of the *k*-means algorithm have proven effective to some extent. However, there is a still a need to run the algorithm multiple times, with different initial centroids, and to average these results to obtain a more stable overall result (Morissette & Chartier, 2013). Unfortunately, there is no method to help determine the number of runs required to guarantee the best results. Moreover, there is no consensus on which variant works the best (Fränti & Sieranoja, 2019).

3. The Proposed Method

As the breaking down of the *m*TSP can be considered a problem of dimensionality reduction in which a few independent clusters are formed from a large number of cities in order to minimize the total intra-cluster distance, it is proposed to use a two-stage method. In the initial stage, clusters of cities are identified using a modified form of factor analysis (FA). This stage is then followed by the use of a simple integer-programming model to allot the cities to the identified clusters. The selection of this method has been inspired by the successful implementation of similar methods in different applications (e.g., Albadawi, Bashir & Chen, 2005; Bashir & Karaa, 2008; Hamdan & Bashir, 2015). It worth noting that in FA terminology, clusters are called factors. Therefore, these two terms will be used interchangeability throughout the rest of the paper.

3.1. Stage 1: Identification of Clusters

FA is a dimensionality reduction technique developed by Spearman in 1904 to extract a few independent factors from the correlation matrix of a large number of interrelated variables using different extraction methods including

canonical factor analysis, common factor analysis, image factor analysis, and principal component analysis (PCA) In this study, PCA was selected as an extraction method because it is straightforward, quantitatively rigorous, and popular (Rummel, 1988). The factors extracted using PCA comprise uncorrelated linear groupings of the initial variables. The variances accounted for by the first few factors typically represent a high percentage of the total variance of the original variables, which means that these variables can be assigned to a few independent factors. However, because the factors obtained using PCA are commonly correlated with many original variables, rotation methods are applied to ensure that each variable is related to only one factor and that each factor is highly correlated with just a small number of variables. A more detailed description of FA can be found in Rummel (1988).

The application of the modified form of FA to identify clusters of cities involves the generation of the matrix of relative distances, the extraction of preliminary clusters, and the obtaining of the final clusters. In this sub-section, a simple hypothetical example is used to explain these steps.

3.1.1. Generation of the relative distance matrix

As mentioned above, the input for FA is a correlation matrix created from an original data set. In the proposed method, this matrix is replaced by one referred to as the relative distance (*RD*) matrix, in which the principal diagonal elements are 1 and every off-diagonal element R_{ij} is defined by equation (1):

$$R_{ij} = 1 - \frac{d_{ij}}{d_{max}} \quad (1)$$

Where d_{ij} is the distance between any pair of cities i and j, and d_{max} is the maximum distance among the cities.

As a demonstration, consider the matrix presented in Table 2, which shows the distances between every pair of cities for a simple hypothetical example of nine cites. Table 3 provides the corresponding *RD* matrix, which was obtained by applying equation (1).

City	1	2	3	4	5	6	7	8	9
1	0	2396	2215	991	3608	1882	1366	1972	1535
2	2396	0	1571	1505	1212	588	3762	1308	3205
3	2215	1571	0	1324	2671	2159	3581	2879	3024
4	991	1505	1324	0	2717	1769	2257	2489	1700
5	3608	1212	2671	2717	0	1726	4974	1636	4417
6	1882	588	2159	1769	1726	0	3248	720	2691
7	1366	3762	3581	2257	4974	3248	0	3338	557
8	1972	1308	2879	2489	1636	720	3338	0	3033
9	1535	3205	3024	1700	4417	2691	557	3033	0

Table 2. Distance matrix for the hypothetical example

City	1	2	3	4	5	6	7	8	9
1	1.00	0.52	0.55	0.80	0.27	0.62	0.73	0.60	0.69
2	0.52	1.00	0.68	0.70	0.76	0.88	0.24	0.74	0.36
3	0.55	0.68	1.00	0.73	0.46	0.57	0.28	0.42	0.39
4	0.80	0.70	0.73	1.00	0.45	0.64	0.55	0.50	0.66
5	0.27	0.76	0.46	0.45	1.00	0.65	0.00	0.67	0.11
6	0.62	0.88	0.57	0.64	0.65	1.00	0.35	0.86	0.46
7	0.73	0.24	0.28	0.55	0.00	0.35	1.00	0.33	0.89
8	0.60	0.74	0.42	0.50	0.67	0.86	0.33	1.00	0.39
9	0.69	0.36	0.39	0.66	0.11	0.46	0.89	0.39	1.00

Table 3. RD matrix for the hypothetical example

3.1.2. Extraction of the preliminary clusters

In this step, equation (2) is used to calculate the eigenvalues and associated eigenvectors (factor loadings) of the *RD* matrix. Per the matrix theory, because it is truly symmetric, this matrix has real eigenvalues and their corresponding eigenvectors are independent.

$$(\lambda_i I - RD)\ Y_i = 0 \quad \forall i = 1, \ldots, n \tag{2}$$

where RD is the relative distance matrix of size $n \times n$, I is the identity matrix, λ_i is the characteristic root (eigenvalue), and Y_i is the corresponding eigenvector.

The eigenvalues calculated for the *RD* matrix (Table 3) are presented in descending order in Table 4. The total variance that every cluster revealed is provided in the first column, which is labelled "eigenvalue." The second column presents the percentage of the total variance attributable to each cluster. The last column, which is the cumulative percentage, shows the percentage of variance attributable to that cluster and those preceding it in the table.

Of these nine clusters, only those with the highest k eigenvalues need to be selected to form preliminary clusters, where k represents the desired number of clusters that need to be formed. For example, if it is decided that three clusters have to be formed, then the clusters that correspond the highest three eigenvalues have to be chosen, and so on.

For the hypothetical example, the formed preliminary clusters are shown in Table 5, where it was assumed that two clusters needed to be formed. These two clusters are the eigenvectors that corresponded to the largest two eigenvalues (5.413 and 1.798) of the *RD* matrix. The elements of the eigenvectors are called loadings and the cities were assigned to the cluster associated with highest absolute loading.

Cluster	Eigenvalue	% of Total variance	% Cumulative percentage (%)
1	5.413	60.15	60.15
2	1.798	19.97	80.12
3	0.734	8.16	88.28
4	0.347	3.85	92.13
5	0.274	3.04	95.17
6	0.208	2.31	97.49
7	0.117	1.30	98.79
8	0.067	0.75	99.54
9	0.042	0.46	100.00

Table 4. The computed eigenvalues

City	Cluster 1	Cluster 2
1	0.358	0.270
2	0.369	-0.302
3	0.317	-0.085
4	0.375	0.115
5	0.272	-0.469
6	0.379	-0.207
7	0.262	0.540
8	0.344	-0.218
9	0.230	0.462

Table 5. Preliminary clusters of the hypothetical example

3.1.3. Obtaining the Final Clusters

Determining which cities should be allocated to each cluster on the basis of the preliminary clusters is not easy in many cases, because a city might have a similar absolute value of loading on more than one cluster. In PCA, this problem is commonly overcome using a rotation technique such as varimax. Kaiser (1960) provides more details about this technique.

Applying the varimax technique to the matrix of preliminary clusters detailed in Table 5 produced the matrix of final clusters presented in Table 6. As shown in Table 6, Cities 1, 4, 7, and 9 had the largest loadings in cluster 1, while Cities 2, 3, 5, 6, and 8 had the largest loadings in cluster 2. Thus, the best grouping for the nine cities was grouping Cities 1, 4, 7 and 9 in one cluster and Cities 2, 3, 5, 6, and 8 in another.

City	Cluster 1	Cluster 2
1	0.432	0.119
2	-0.014	0.477
3	0.126	0.303
4	0.320	0.227
5	-0.206	0.502
6	0.067	0.426
7	0.588	-0.122
8	0.037	0.406
9	0.549	-0.045

Table 6. Final clusters of the hypothetical example

3.2. Stage 2: Assigning Cities to Clusters

If the problem is small, then employing the preceding steps is enough to form the clusters. However, assigning the cities to clusters manually is impractical for large problems. This task can be facilitated using a simple binary integer programming model, such as the following:

Maximize

$$\sum_{i=1}^{n}\sum_{u=1}^{k} w_{iu}{}^{2} x_{iu} \tag{3}$$

Subject to

$$\sum_{u=1}^{k} x_{iu} = 1 \qquad \forall \; i = 1, \ldots, n \tag{4}$$

Where k is the number of clusters, n is the total number of cities, w_{iu} is the loading of city i on cluster u, and x_{iu} is 1 if a city i is allotted to cluster u and 0 if not.

Realistic constraints can be easily incorporated in this model. For instance, if the maximum number of cities that can be visited by a salesman in a cluster is bounded, then the following constraint can be added:

$$\sum_{i=1}^{n} x_{iu} \leq l \qquad \forall \; u = 1, \ldots, k \tag{5}$$

where l is the maximum number of cities a salesman can visit in cluster u.

4. Performance Evaluation

This section provides a summary of the evaluation of the performance of the proposed method with regard not only to the quality of formed clusters, but also to the extent that the method improves the performance of heuristics optimization algorithms in solving the *m*TSP. This evaluation was conducted by solving test problems without considering any constraints, using two combinations: the method that we proposed with a genetic algorithm (FA-GA) and the *k*-means++ with a genetic algorithm (KM-GA).

The main reason for comparing our method with the *k*-means++ was because it outperforms the standard *k*-means (Arthur & Vassilvitskii, 2007), which was used in most studies proposing clustering-based optimization approaches to solve the *m*TSP (see Table 1). The difference between these two algorithms is in their initialization methods. The following steps summarize the *k*-means++ initialization method:

1. One centroid, c_1, is selected randomly from the set of data, X.
2. For each data point $x \in X$, the distance d(x) to the closest selected centroid is computed.
3. A new centroid c_i is selected using a weighted probability defined by $\frac{d(x)^2}{\sum d(x)^2}$.
4. Steps 2 and 3 are repeated until the centroids for all clusters are identified.
5. The steps of the standard *k*-means algorithm are then followed.

To evaluate to the extent to which the proposed clustering method improves the performance of heuristics optimization algorithms in solving the *m*TSP, it was necessary to combine the proposed method with a selected heuristics optimization algorithm and compare the performance of this combination with that of a combination of the *k*-means++ and the same selected heuristics optimization algorithm. Because it is beyond the scope of this study to propose a heuristics optimization algorithm to determine the optimal routes, a basic genetic algorithm (an open-source MATLAB code provided by Kirk (2014) was used. The major steps of this algorithm are as follows:

1. Create a starting population of 80 solutions.
2. Iterate the following steps:
   a. Evaluate the cost of the current population of solutions;
   b. Identify the best solution in the population; and
   c. Modify the population by randomly putting them in groups of eight and performing mutations on seven of the eight, while keeping the best of the eight to pass along to the next generation.
3. Use the best solution found.

4.1. Evaluation Criteria

Four objective criteria that are commonly utilized in the literature were chosen for performance evaluation. These are: 1) the sum of the squared error; 2) the variability of cluster size; 3) the total traveling distance; and 4) the running time. The first two criteria were used to evaluate performance with respect to the obtained clustering solutions, whereas the other two criteria were used to evaluate performance with respect to the optimal route solutions obtained.

4.1.1. The Sum of the Squared Error

The Sum of the Squared Error (SSE) is commonly used to measure intra-cluster variance, which is the most uncomplicated and commonly utilized criterion for measuring the quality of clustering. For the purposes of this study, SSE is the sum of the squared distances between the coordinates of the points (cities) and the coordinates of centroids of the corresponding clusters (Cao, Liang & Jiang, 2009). It is calculated as:

$$\text{SSE} = \sum_{i \in u} \sum_{r=1}^{N} (c_{ir} - C_{ur})^2 \qquad \forall \quad u = 1, \ldots, k \tag{6}$$

Where C_{ur} is the coordinate of cluster u centroid along dimension r, c_{ir} is the coordinate of city i along dimension r, k is the number of clusters, and N is the number of dimensions. According to equation (6), SSE decreases as the

number of clusters grows. This is because an increase in the number of clusters will lead to a decrease in the sizes of the clusters, and thus a reduction in SSE value.

4.1.2. Variability of Cluster Size

As mentioned earlier, one common limitation of the *k*-means is the formation of unbalanced or even empty clusters (Kaur & Kaur, 2013; Morissette & Chartier, 2013; Singh et al., 2012). This limits its practicality for some applications of the *m*TSP, such as sales territories assignment. For instance, according to Bolaños, Echeverry and Escobar (2015), a balanced workload between clusters is critical due to its impact on the salespeople's morale, satisfaction, and likelihood of achieving their targets. One simple method to evaluate the variability of cluster size (V) is standard deviation, defined by equation (7):

$$V = \sqrt{\frac{\sum_{u=1}^{k}(S_u - \bar{S})^2}{k-1}} \tag{7}$$

Where k is the number of clusters, $\bar{S}$ is the average cluster size, and S_u is the size of cluster u (i.e., the number of cities in cluster u).

4.1.3. Total Traveling Distance

A common objective of any traveling salesman problem is the minimization of the total traveling distance (Bolaños et al., 2015), so that the total traveling cost is also minimized. Equation (8) defines the total traveling distance (TTD):

$$\text{TTD} = \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}T_{ij} \tag{8}$$

where d_{ij} is the distance between cities i and j, n is the number of cities, and $T_{ij} = 1$ if a salesman is traveling directly from city i to j in the optimal route and 0 if not.

4.1.3. Running Time

Running time, defined as the total CPU time (in seconds) spent during the execution of an algorithm, is one of the most critical performance criteria. Because it typically grows with input size and other factors, it can be considered a measure of the practicality of an algorithm. In this study, comparisons were made between the two combinations (FA-GA and KM-GA) with respect to the impacts of both problem size and the number of clusters according to the running time.

4.2. Test Problems

The two combinations, FA-GA and KM-GA, were tested using seven standard instances obtained from the TSPLIB library (https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/). As shown in Table 7, the sizes of these instances ranged from 52 to 1002 cities. For every instance, an additional city representing the depot was added to each instance. The average of the coordinates of the cities in each instance were considered the coordinates of the added depot. Each of the seven instance was solved using nine different numbers of salesmen, ranging from two to 10.

No.	Instance	No. of Cities
1	Berlin52	52
2	Eil76	76
3	KorA100	100
4	KorA200	200
5	Lin318	318
6	Pr439	439
7	Pr1002	1002

Table 7. Test problems

4.3. Number of Runs

One advantage of the method proposed in this study is that it does not require multiple runs. In contrast, the *k*-mean++ algorithm begins by randomly selecting the location of the initial centroid, so selecting different starting centroids can result in different clusters being formed. The *k*-means++ algorithm must therefore be run several times (Morissette & Chartier, 2013). In the current study, the *k*-means++ algorithm was run independently 20 times to solve each test problem. Accordingly, the performance evaluation of the *k*-means++ in terms of each criterion was based on the average values of the results of 20 runs. Therefore, a total of 1,323 runs were performed in this study (1,260 runs of KM-GA and 63 of FA-GA).

5. Performance Evaluation Results

Table A-1 in the appendix presents a summary of the performance evaluation results using MATLAB 2018 on a system with a CPU Intel core i7.

5.1. Quality of Clusters

As a sample of the solutions, the obtained optimal route for each salesman —for instance, KorA100 ($k = 6$)— is presented in Table 8. A general observation can be made about the formed clusters for all the test problems, including KorA100, whose formed clusters are displayed in Figures 1 and 2: FA-GA generates clusters with non-crossed paths, whereas KM-GA generates clusters with crossed paths. This feature makes FA-GA more practical in applications that require avoiding the crossing of paths, such as multi-robot task allocation and unmanned aerial vehicles. The reason that FA-GA generates exclusive, distinctive clusters is due to its mathematical basis, i.e., the orthogonality of the eigenvectors of the matrix of relative distances.

Figures A-1 and A-2 in the appendix are the plots of k values versus the SSE values of the solutions that the two combinations, FA-GA and KM-GA, produced for KorA200 and P439. A visual assessment of these plots as well as those obtained for the other test problems shows that regardless of the k values, FA-GA significantly outperformed KM-GA with regard to minimizing the SSE. It is also clear that the SSE of FA-GA generally decreased quickly until a particular value of k was reached; after this point the values continued to decrease, but at a reduced rate. Conversely, the relationship between SSE of KM-GA and k did not follow a particular pattern. One conceivable interpretation for this behavior is that each plotted value of SSE was an average value of SSEs obtained over 20 runs. It is worth noting that, as shown in Table A-1 in the appendix, these findings are applicable to the solutions obtained not only for KorA200 and P439 but also those obtained for all the other instances.

Cluster (salesman)	FA-GA		KM-GA	
	Number of cities	Route	Number of cities	Route
1	18	0 60 62 35 86 27 20 12 55 83 34 7 9 57 87 51 61 58 0	13	0 25 81 85 68 73 50 44 2 54 40 64 69 0
2	16	0 4 65 26 18 66 70 22 16 88 94 24 38 79 53 90 0	22	0 91 32 11 17 15 45 23 77 60 62 35 86 27 12 20 57 7 9 87 51 58 0
3	17	0 64 40 54 2 44 73 50 82 95 76 33 13 85 68 81 25 0	11	0 96 78 52 5 37 33 76 13 95 82 0
4	16	0 39 37 5 96 78 52 48 100 41 71 14 3 43 46 29 0	13	0 48 100 41 71 14 3 43 46 29 34 83 55 0
5	17	0 72 10 84 36 99 59 74 21 17 15 11 32 45 91 98 23 0	24	0 79 53 19 4 65 26 18 66 70 22 16 88 94 24 38 99 36 84 10 72 21 74 59 0
6	17	0 28 1 63 6 49 75 97 56 80 31 89 42 8 92 67 101 0	17	0 8 42 89 31 80 56 97 75 92 49 6 63 1 47 93 28 0

City 0 represents the depot.

Table 8. Optimal routes for instance KorA100

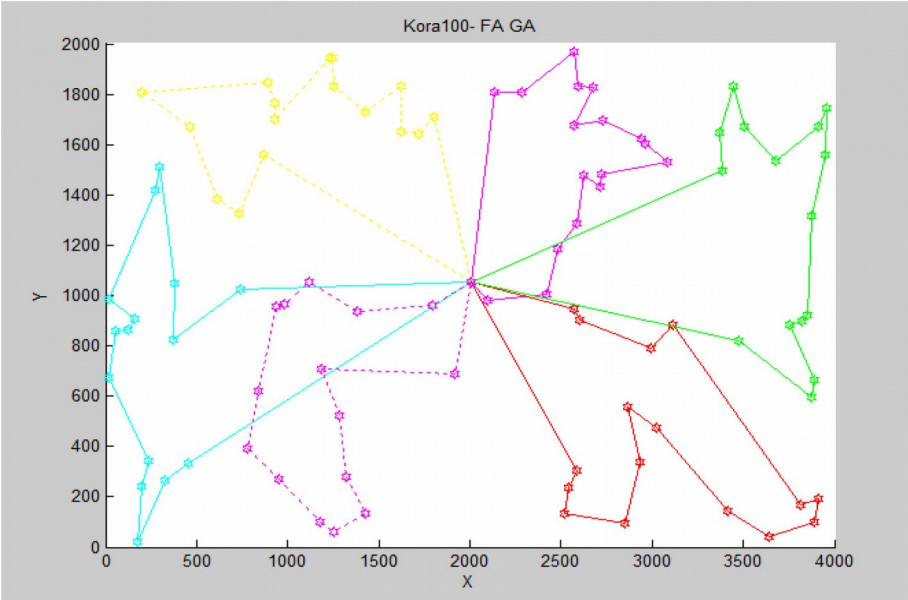

Figure 1. Clusters formed for the instance KorA100 by FA-GA (MATLAB output)

With regard to the variability of cluster size, the results presented in Table A-1 indicate that, compared to those produced by KM-GA, the clusters produced by FA-GA had a lower cluster size variability in 65 percent of the 63 solved problems. Thus, FA-GA offers the advantage of achieving a good workload balance among the salesmen.

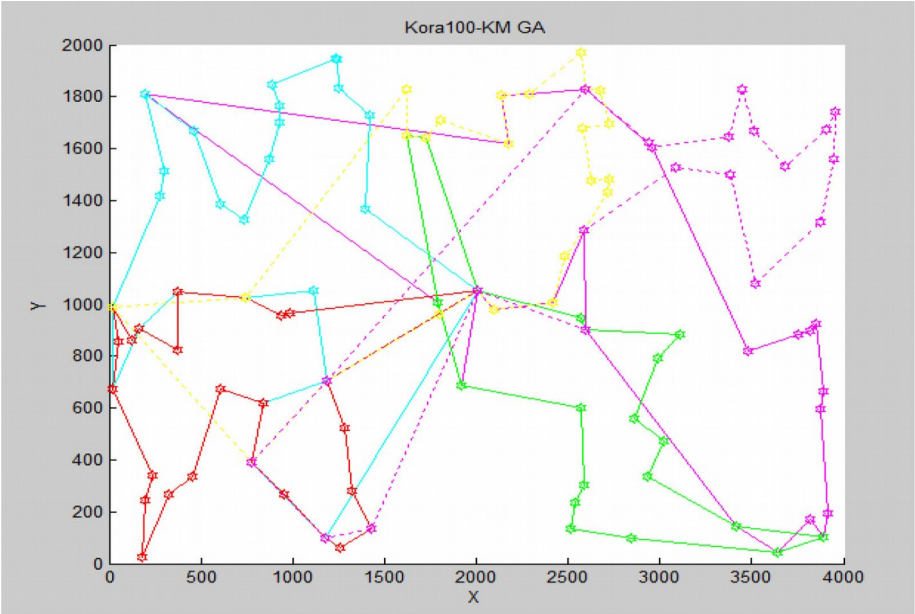

Figure 2. Clusters formed for instance KorA100 by KM-GA (MATLAB output)

5.2. Impact on Finding Optimal Routes

Figures A-3 and A-4 in the appendix illustrate the plots of k values versus the total traveling distance of the solutions produced by FA-GA and KM-GA for KorA100 and Pr1002. A visual observation of these plots and those obtained for the other instances indicates that when the number of clusters grows, so too does the total traveling distance. This observation is applicable to both combinations. Furthermore, as shown in Table A-1 in the appendix, FA-GA plainly outperformed KM-GA with respect to minimizing the total traveling distance for all instances, regardless of the number of clusters.

Figures A-5 and A-6 in the appendix show plots of instance size versus the total CPU running time in seconds for the solutions produced using the two combinations for the seven instances of the nine different values of *k*. A visual assessment of these plots indicates that FA-GA obviously performed better than KM-GA with regard to the insensitivity of the total CPU running time to the problem size. Moreover, as shown in Table A-1 in the appendix, compared to FA-GA, KM-GA used less CPU running time in 22 percent of the solutions. However, these solutions required longer traveling distances than those produced using FA-GA.

6. Summary and Conclusion

The *m*TSP, an NP-hard problem, is among the most widely discussed problems in combinatorial optimization and several procedures to solve it have been proposed in the literature. Among these procedures is the use of a clustering-based optimization approach in which a clustering algorithm is used to break the problem down into a number of TSPs equal to the number of salesmen, and then the tour for each TSP is determined using an optimization algorithm. To break down the *m*TSP, this study proposed a new method comprising two stages. First, the clusters are identified via a modified form of FA, and then a simple integer programming model is used for the assignment of cities to the identified clusters, considering realistic constraints such as the maximum number of cities to be assigned to each salesman.

To evaluate the performance of the proposed method, it was combined with Kirk's (2014) genetic algorithm to solve a total of 63 test problems (seven standard instances, each solved nine times by varying the number of clusters from two to 10). These solutions were then compared to those produced by combining the *k*-means++ and Kirk's genetic algorithm. The comparison of the results demonstrated that the combination that included the proposed method compared favorably to the other based on four evaluation criteria: 1) the sum of the squared error; 2) the variability of cluster size; 3) total traveling distance; and 4) running time.

This study makes a twofold contribution to the literature. First, unlike previous studies, it tackles the issue of improving the performance of clustering-based optimization approaches in solving the *m*TSP by proposing a method that produces better cluster solutions rather than by proposing a new or improved version of an optimization algorithm for finding optimal routes. Second, this first study to use a FA-based method to break down the *m*TSF. In addition to its superior performance compared to the *k*-mean++ and in producing distinct clusters, this method does not require random selection of the initial centroids of clusters. Thus, an important advantage of the proposed method is its stability compared to the *k*-mean++ and other variants of the *k*-means that may provide different clusters for each run.

Although this study represents useful progress, much remains to be done. For instance, the performance of the proposed method was evaluated using 63 problems (seven instances, each of which was solved with nine different numbers of salesmen). However, there is a need to provide more evidence of the robustness of the proposed method by conducting an extensive computational study using instances of larger sizes. Moreover, combining the proposed method with Kirk's (2014) genetic algorithm to find the optimal routes was only for comparison purposes. To find optimal routes, future studies could explore combining the proposed method with new, current, or improved versions of heuristics optimization algorithms (such as those presented in Table 1).

Declaration of Conflicting Interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

References

Albadawi, Z., Bashir, H.A., & Chen, M. (2005). A mathematical approach for the formation of manufacturing cells. *Computers and Industrial Engineering,* 21(1), 3-21. https://doi.org/10.1016/j.cie.2004.06.008

Ann, S., Kim, Y., & Ahn, J. (2015). Area allocation algorithm for multiple UAVs Area coverage based on clustering and graph method. *IFAC-PapersOnLine,* 48(9), 204-209. https://doi.org/10.1016/j.ifacol.2015.08.084

Arthur, D., & Vassilvitskii, S., (2007). K-Means++: The Advantages of careful seeding. Proceedings of the *8th annual ACM-SIAM symposium on Discrete algorithms* (8, 1027-1025). https://doi.org/10.1145/1283383.1283494

Bashir, H.A., & Karaa, S. (2008). Assessment of clustering tendency for the design of cellular manufacturing systems. *Journal of Manufacturing Technology Management,* 19(8), 1004-1014. https://doi.org/10.1108/17410380810911754

Bektas, T., (2006). The multiple traveling salesman problem: An overview of formulations and solution procedures. *The International Journal of Managemnt Science,* 34(3), 209-219. https://doi.org/10.1016/j.omega.2004.10.004

Bolaños, R.I., Echeverry, M.G., & Escobar, J.W., (2015). A multiobjective non-dominated sorting genetic algorithm (NSGA-II) for the multiple traveling salesman problem. *Decision Science Letters,* 4(4), 559-568. https://doi.org/10.5267/j.dsl.2015.5.003

Boone, N., Sathyan, A., & Cohen, K. (2015). Enhanced approaches to solving the multiple traveling. *In American Institute of Aeronautics and Astronautics,* 1-7. https://doi.org/10.2514/6.2015-0889

Cao, F., Liang, J., & Jiang, G. (2009). An initialization method for the k-means algorithm using neighborhood model. *Computers and Mathematics with Applications,* 58(3), 474-483. https://doi.org/10.1016/j.camwa.2009.04.017

Carter, A.E., & Ragsdale, C.T. (2006). A new approach to solving the multiple traveling salesperson problem using genetic algorithms. *European Journal of Operational Research,* 175(1), 246-257. https://doi.org/10.1016/j.ejor.2005.04.027

Chandran, N., Narendrananesh, K., & Ganesh, K. (2006). A clustering approach to solve the multiple travelling salesmen problem. *International Journal of Industrial and Systems Engineering,* 1(3), 372-387. https://doi.org/10.1504/IJISE.2006.009794

Elango, M., Nachiappan, S., & Tiwari, M.K. (2011). Balancing task allocation in multi-robot systems using k-means clustering and auction based mechanisms. *Expert Systems with Applications,* 38, 6486-6491. https://doi.org/10.1016/j.eswa.2010.11.097

Expósito-izquierdo, C., Rossi, A., & Sevaux, M. (2016). A two-level solution approach to solve the clustered capacitated vehicle routing problem. *Computers & Industrial Engineering,* 91, 274-289. https://doi.org/10.1016/j.cie.2015.11.022

Forgy, E. (1965) Cluster analysis of multivariate data: efficiency vs. interpretability of classification. *Biometric,* 21, 768-769.

Fränti, P., & Sieranoja, S. (2019). How much can k-means be improved by using better initialization and repeats? *Pattern Recognition,* 93, 95-112. https://doi.org/10.1016/j.patcog.2019.04.014

Ghafurian, S., & Javadian, N. (2011). An ant colony algorithm for solving fixed destination multi-depot multiple traveling salesmen problems. *Applied Soft Computing,* 11(1), 1256-1262. https://doi.org/10.1016/j.asoc.2010.03.002

Gorenstein, S. (1970). Printing press scheduling for multi-edition periodicals. *Management Science,* 16(6), 373-383. https://doi.org/10.1287/mnsc.16.6.B373

Hamdan, B.I., & Bashir, H. (2015). A two-stage decomposition approach for the traveling salesman problem. Proceedings of the *IEOM 5th International Conference on Industrial Engineering and Operations Management.* https://doi.org/10.1109/IEOM.2015.7093868

Harrath, Y., Salman, A.F., Alqaddoumi, A., Hasan, H., & Radhi, A. (2019). A novel hybrid approach for solving the multiple traveling salesmen problem. *Arab Journal of Basic and Applied Sciences,* 26(1), 103-112. https://doi.org/10.1080/25765299.2019.1565193

Kaiser, H.F. (1960). The application of electronic computers to factor analysis. *Educational and Psychological Measurement,* 20(1), 141-151. https://doi.org/10.1177/001316446002000116

Kaur, S., & Kaur, U. (2013). A survey on various clustering techniques with k-means clustering algorithm in detail. *International Journal of Computer Science and Mobile Computing,* 2(4), 155-159.

Király, A., & Abonyi, J. (2010). A novel approach to solve multiple traveling salesmen problem by genetic algorithm. *Studies in Computational Intelligence,* 141-151. https://doi.org/10.1007/978-3-642-15220-7_12

Kirk, J. (2014). *Multiple traveling salesmen problem - genetic algorithm.* MATLAB Central File Exchange.

Kitjacharoenchai, P., Ventresca, M., Moshref-Javadi, M., Lee, S., Tanchoco, J.M.A., & Brunese, P.A. (2019). Multiple traveling salesman problem with drones: mathematical model and heuristic approach. *Computers & Industrial Engineering,* 129, 14-30. https://doi.org/10.1016/j.cie.2019.01.020

Kovács, L., Agárdi, A., & Debreceni, B. (2018). Efficiency analysis of the vertex clustering in solving the traveling salesman problem. *Annales Mathematicae et Information,* 48(1), 33-42.

Larki, H., & Yousefikhoshbakht, M. (2014). Solving the multiple traveling salesman problem by a novel meta-heuristic algorithm. *Journal of Optimization in Industrial Engineering,* 7(16), 55-63.

Li, M., Xu, D., Yue, J., Zhang, D. & Zhang, P. (2020). The seeding algorithm for k-means problem with penalties. *Journal of Combinatorial Optimization,* 39, 15-32. https://doi.org/10.1007/s10878-019-00450-w

Li, M., Xu, D., Zhang, D. & Zhou, H., (2020). The provably good parallel seeding algorithms for the k-means problem with penalties. *International Transactions in Operational Research,* 00, 1-14. https://doi.org/10.1111/itor.12808

Lin, S., & Kernighan, B.W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research,* 21(2), 498-516.

Liu, W., Li, S., & Zhao, F. (2009). An ant colony optimization algorithm for the multiple traveling salesmen problem. Proceedings of the *4th IEEE Conference on Industrial Electronics and Applications* (1533-1537).

Lu, Z., Zhang, K., He, J., & Niu, Y. (2016). Applying k-means clustering and genetic algorithm for solving mTSP. Proceedings of the *International Conference on Bio-Inspired Computing: Theories and Applications* (278-284). https://doi.org/10.1007/978-981-10-3614-9

Ma, J., Shi, S., & Gu, X., (2018). An optimization-based mTSP clustering algorithm for wireless sensor networks. Proceedings of *14th International Wireless Communications & Mobile Computing Conference (IWCMC)* (334-338). IEEE. https://doi.org/10.1109/IWCMC.2018.8450272

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. Proceedings of the *Fifth Berkeley Symposium on Mathematical Statistics and Probabilities* (1, 281-296).

Mardiyati, S., Safitri, L., & Jihan., (2017). Solving multiple traveling salesman problem using k-means clustering-genetic ant aolony system algorithm. *Far East Journal of Mathematical Sciences,* 102(7), 1417-1432. https://doi.org/10.17654/MS102071417

Modares, A., Somhom, S., & Enkawa., T. (1999). A self-organizing neural network approach for multiple traveling salesman and vehicle routing problems. *International Transactions in Operational Research,* 6(6), 591-606. https://doi.org/10.1111/j.1475-3995.1999.tb00175.x

Morissette, L., & Chartier, S. (2013). The k-means clustering technique: general considerations and implementation in mathematica. *Tutorials in Quantitative Methods for Psychology,* 9(1), 15-24. https://doi.org/10.20982/tqmp.09.1.p015

Nallusamy, R., Duraiswamy, K. Dhanalaksmi, R., & Parthiban, P. (2009a). Optimization of multiple vehicle routing problems using approximation algorithms. *International Journal of Engineering Science and Technology,* 1(3), 129-135.

Nallusamy, R., Duraiswamy, K., Dhanalaksmi, R., & Parthiban, P. (2009b). Optimization of non-linear multiple traveling salesman problem using k-means clustering , shrink wrap algorithm and meta-heuristics. *International Journal of Nonlinear Science,* 8(4), 480-487. https://doi.org/10.26634/jse.4.2.1069

Necula, R., Breaban, M., & Raschip, M. (2015). Performance evaluation of ant colony systems for the single-depot multiple traveling salesman problem. In Lecture Notes in *Artificial Intelligence (Subseries of Lecture Notes in Computer Science),* 257-268. https://doi.org/10.1007/978-3-319-19644-2_22

Paydar, M., Mahdavi, I., Sharafuddin, I., & Solimanpur, M. (2010). Applying simulated annealing for designing cellular manufacturing systems using mdmtsp. *Computers & Industrial Engineering,* 59(4), 929-936. https://doi.org/10.1016/j.cie.2010.09.003

Rummel, R.J. (1988). *Applied factor analysis.* Evanston: Northwestern University Press.

Russell, R.A. (1977). An effective heuristic for the m-tour traveling salesman problem with some side conditions. *Operations Research,* 25(3), 517-524. https://doi.org/10.1287/opre.25.3.517

Sadiq. S. (2012). The Traveling Salesman Problem: Optimizing Delivery Routes Using Genetic Algorithms. *SAS Global Forum.*

Saleh, H.A., & Chelouah, R. (2004). The design of the global navigation satellite system surveying networks using genetic algorithms. *Engineering Applications of Artificial Intelligence,* 17(1), 111-122. https://doi.org/10.1016/j.engappai.2003.11.001

Sedighpour, M., Yousefikhoshbakht, M., & Darani, N.M. (2011). An effective genetic algorithm for solving the multiple traveling salesman problem. *Journal of Optimization in Industrial Engineering,* 4(8), 73-79.

Shabanpour, M., Yadollahi, M. & Hasani, M.M. (2017). A new method to solve the multi traveling salesman problem with the combination of genetic algorithm and clustering technique. *International Journal of Computer Science and Network Security*, 17(5), 221-230.

Singh, A., & Baghel, A.S. (2009). A new grouping genetic algorithm approach to the multiple traveling salesperson problem. *Soft Computing,* 13(1), 95-101. https://doi.org/10.1007/s00500-008-0312-1

Singh, K., Malik, D., & Sharma, N. (2012). Evolving limitations in k-means algorithm in data mining and their removal. *International Journal of Computational Engineering & Management,* 12, 105-109.

Snyder, L.V., & Daskin, M.S. (2006). A random-key genetic algorithm for the generalized traveling salesman problem. *European Journal of Operational Research,* 174(1), 38-53. https://doi.org/10.1016/j.ejor.2004.09.057

Sofge, D., Schultz, A., & De Jong, K. (2002). Evolutionary computational approaches to solving the multiple traveling salesman problem using a neighborhood attractor schema. Proceedings of the *Applications of Evolutionary Computing on EvoWorkshops* (153-162).

Somhom, S., Modares, A., & Enkawa, T. (1999). Competition-based neural network for the multiple travelling salesmen problem with minmax objective. *Computers and Operations Research,* 26(4), 395-407. https://doi.org/10.1016/S0305-0548(98)00069-0

Sze, S. N., & Tiong, W.K., (2007). A comparison between heuristic and meta- heuristic methods for solving the multiple traveling salesman problem. *International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering,* 1(1), 27-30.

Tang, L., Liu, J., Rong, A., & Yang, Z. (2000). A multiple traveling salesman problem model for hot rolling scheduling in shanghai baoshan iron & steel complex. *European Journal of Operational Research,* 124(2), 267-282. https://doi.org/10.1016/S0377-2217(99)00380-X

Trigui, S., & Koub, A. (2016). A clustering market-based approach for multi-robot emergency response applications. Proceedings of the *International Conference on Autonomous Robot Systems and Competitions* (137-143). https://doi.org/10.1109/ICARSC.2016.14

TSPLIB (n.d.). https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/

Yu, Z., Jinhai, L., Guochang, G.U., Rubo, Z., & Haiyan, Y. (2002). An implementation of evolutionary computation for path planning of cooperative mobile robots. *Proceedings of the 4th World Congress on Intelligent Control and Automation (1798–1802).*

Yuan, S., Skinner, B., Huang, S., & Liu, D. (2013). A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms. *European Journal of Operational Research,* 228(1), 72-82. https://doi.org/10.1016/j.ejor.2013.01.043

Zeebaree, D.Q., Haron, H., Abdulazeez, A.M., & Zeebaree, S.R.M. (2017). Combination of k-means clustering with genetic algorithm: A review. *International Journal for Applied Engineering Research,* 12(24), 14238-14245.

Appendix

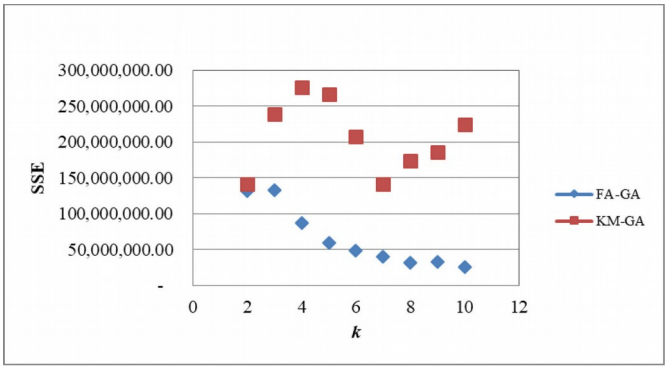

Figure A-1. Sum of the squared error vs. *k* values for instance KorA200

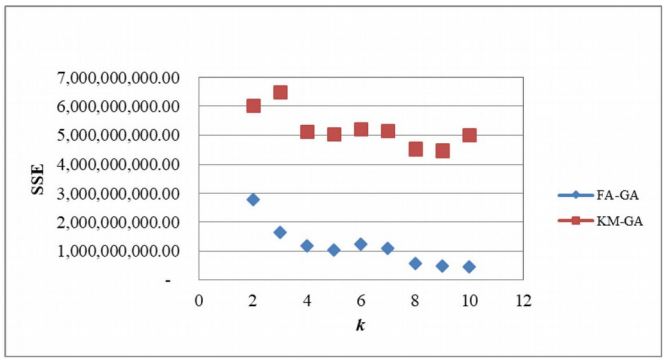

Figure A-2. Sum of the squared error vs. *k* values for instance P439

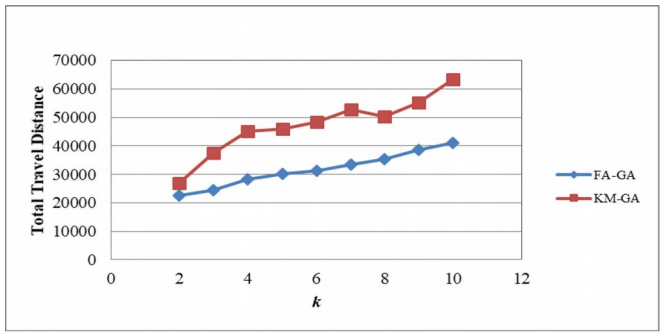

Figure A-3. Total traveling distance vs. *k* for instance KorA100

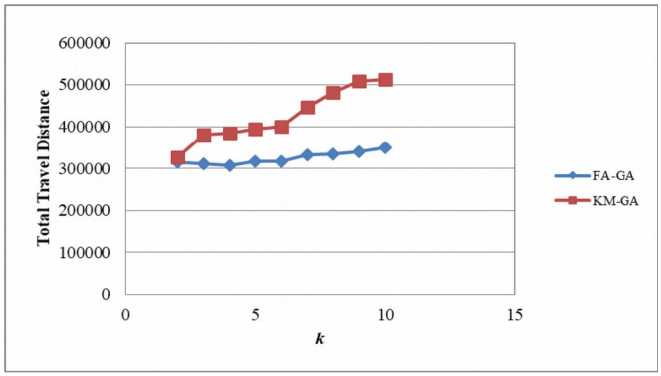

Figure A-4. Total traveling distance vs. *k* for instance Pr1002

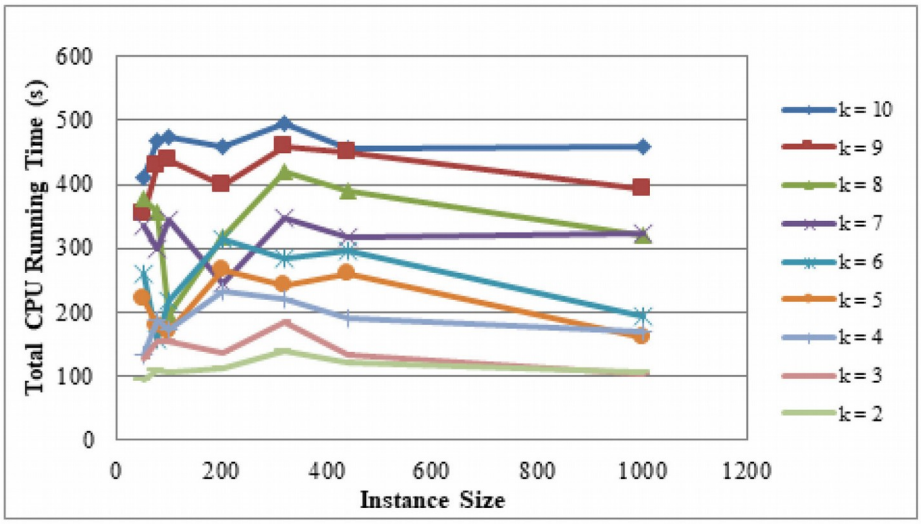

Figure A-5. Total CPU running time vs. instance size for FA-GA

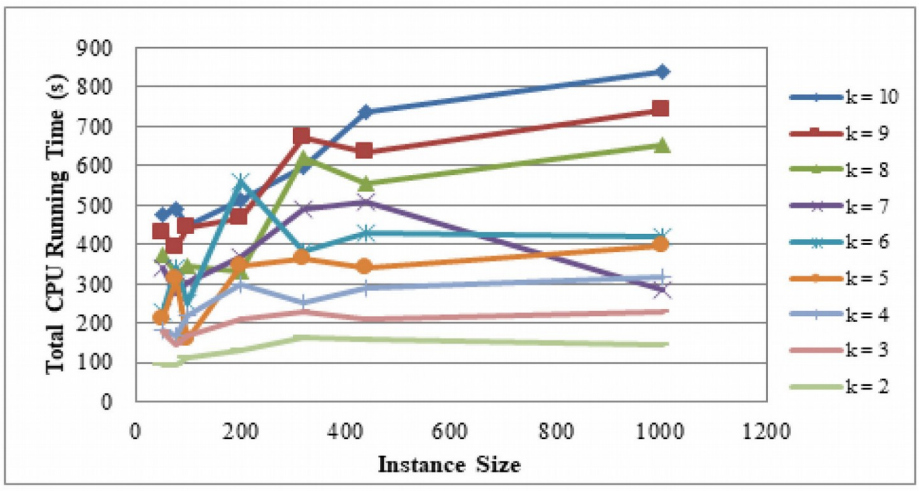

Figure A-6. Total CPU running time vs. instance size for KM-GA

Instance	*k*	Sum of the Squared Error		Variability of Cluster Size		Total Traveling Distance		Running Time	
		FA-GA	KM-GA	FA-GA	KM-GA	FA-GA	KM-GA	FA-GA	KM-GA
Berlin52	2	**6,623,069.89**	13,267,848.08	**7.78**	23.33	**8,134.60**	12,489.55	94.61	**94.06**
	3	**4,070,234.33**	10,904,308.67	**7.64**	10.02	**8,709.76**	12,485.62	**125.85**	179.32
	4	**2,817,398.84**	9,831,700.49	**4.99**	7.89	**9,576.01**	13,281.60	132.39	**54.96**
	5	**2,993,893.40**	9,528,033.95	**4.39**	5.94	**10,998.83**	14,071.93	219.52	**210.70**
	6	**1,918,932.68**	9,163,874.30	5.12	**4.96**	**11,397.21**	15,736.77	260.90	**229.53**
	7	**1,504,664.68**	9,484,157.99	**4.04**	4.86	**11,704.04**	16,825.25	**334.87**	338.89
	8	**1,330,291.88**	9,360,040.27	4.07	**3.02**	**13,064.18**	18,143.91	376.73	**373.02**
	9	**1,087,172.16**	8,678,571.04	**2.80**	3.06	**13,622.35**	17,576.62	**353.00**	428.09
	10	**794,579.54**	8,271,247.22	**2.91**	3.13	**14,271.90**	18,815.96	**409.53**	477.83
Eil76	2	**31,517.10**	35,012.61	3.54	3.54	**594.95**	638.40	109.72	**93.82**
	3	**23,269.24**	28,650.10	2.08	**0.58**	**669.84**	687.90	155.50	**145.68**
	4	**14,148.92**	21,558.98	3.86	**2.65**	**629.05**	714.53	187.57	**170.17**
	5	**10,867.48**	24,640.62	**2.88**	3.58	**700.56**	822.35	**179.81**	312.49
	6	**9,297.12**	27,489.92	**1.72**	2.79	**726.91**	911.25	**158.31**	339.39
	7	**8,099.82**	29,417.47	2.00	**1.91**	**760.44**	1,007.78	299.82	**275.81**
	8	**7,324.63**	28,723.92	2.62	**2.20**	**829.71**	1,087.69	357.41	**325.51**
	9	**6,209.04**	29,742.23	1.59	**1.51**	**13,622.23**	17,576.26	427.38	**393.26**
	10	**5,539.12**	28,278.91	**1.64**	2.67	**889.01**	1,207.64	**467.54**	488.44
KorA100	2	**67,364,536.07**	84,648,635.23	2.12	**1.41**	**22,435.01**	26,857.16	**106.58**	113.97
	3	**63,085,343.39**	138,648,191.61	**6.81**	13.05	**24,458.27**	37,532.20	**154.60**	167.55
	4	**47,492,101.08**	166,694,288.28	**2.87**	5.29	**28,384.52**	45,056.44	**172.21**	220.34
	5	**31,985,965.83**	117,690,996.24	**2.17**	4.36	**30,199.53**	45,975.59	168.72	**157.96**
	6	**23,360,049.09**	119,324,875.23	**0.75**	5.32	**31,186.81**	48,321.49	**217.81**	254.16
	7	**18,754,586.29**	118,602,555.36	**2.51**	2.75	**33,521.43**	52,618.13	343.29	**303.73**
	8	**15,440,702.78**	91,268,325.49	3.02	**2.07**	**35,255.04**	50,253.96	**196.76**	347.35
	9	**13,308,332.30**	100,381,335.88	**1.20**	3.92	**38,590.13**	55,203.61	**436.83**	443.87
	10	**13,284,269.33**	113,705,162.35	2.73	**2.40**	**41,020.95**	63,443.10	474.43	**446.44**
KorA200	2	**131,378,301.88**	140,955,782.76	**0.71**	2.12	**31,899.78**	35,361.22	**112.07**	131.97
	3	**133,452,353.75**	239,131,218.71	**7.81**	11.14	**33,979.67**	47,411.05	**136.47**	210.06
	4	**87,552,817.97**	276,052,069.27	5.32	**3.59**	**36,184.64**	56,009.86	**231.46**	296.72
	5	**59,991,608.48**	266,008,910.90	5.07	**4.32**	**38,421.78**	60,509.30	**266.67**	345.91
	6	**48,841,367.08**	207,087,089.51	**1.05**	5.96	**40,606.38**	60,165.38	**314.86**	559.63
	7	**40,218,581.16**	141,188,975.93	5.22	**4.64**	**42,492.98**	55,356.63	**243.61**	368.84
	8	**32,400,991.15**	174,282,604.40	5.51	**4.85**	**43,473.00**	68,163.17	**315.85**	329.35
	9	**33,475,293.33**	186,037,182.34	**3.61**	4.42	**46,771.67**	67,529.92	**398.46**	465.01
	10	**25,361,126.70**	224,313,951.00	**3.14**	4.38	**48,596.59**	77,932.70	**459.82**	511.82

Instance	k	Sum of the Squared Error		Variability of Cluster Size		Total Traveling Distance		Running Time	
		FA-GA	KM-GA	FA-GA	KM-GA	FA-GA	KM-GA	FA-GA	KM-GA
Lin318	2	**352,586,832.98**	436,048,027.80	13.44	13.44	**47,391.01**	54,775.51	**137.83**	161.82
	3	**274,655,225.26**	424,094,353.00	**15.31**	25.11	**47,890.01**	60,597.78	183.86	**54.26**
	4	**171,129,994.59**	395,439,561.84	**6.24**	13.60	**47,875.83**	65,493.68	**220.87**	252.27
	5	**137,341,184.05**	3,514,00,559.14	14.58	**9.36**	**53,391.30**	71,482.88	**242.30**	363.74
	6	**105,449,069.30**	290,431,838.10	**5.85**	10.61	**56,358.74**	74,190.92	**285.02**	383.96
	7	**87,770,046.06**	349,375,921.00	**7.66**	8.96	**55,728.46**	79,173.04	**348.76**	489.08
	8	**76,517,133.43**	488,339,750.44	**6.06**	11.24	**56,051.38**	95,741.75	**420.20**	621.76
	9	**70,186,785.23**	463,695,884.20	**4.56**	10.14	**58,725.45**	92,253.00	**458.75**	672.34
	10	**54,493,581.90**	436,035,518.09	**6.10**	13.33	**62,559.04**	92,557.06	**496.19**	597.95
Pr439	2	**2,777,264,165.25**	6,033,848,491.14	**80.61**	166.17	**127,318.18**	178,433.77	**120.30**	158.40
	3	**1,651,920,267.48**	6,503,335,649.66	**70.55**	105.19	**125,107.92**	215,592.56	**133.51**	209.18
	4	**1,185,072,944.58**	5,127,375,754.15	31.86	**31.21**	**130,867.32**	195,348.94	**189.04**	288.19
	5	**1,034,224,066.81**	5,021,211,011.23	**18.01**	25.55	**138,537.67**	208,645.95	**260.69**	339.59
	6	**1,239,728,327.44**	5,203,918,594.73	**25.36**	30.22	**153,724.58**	213,950.60	**296.39**	431.25
	7	**1,080,832,676.31**	5,145,426,302.97	**24.51**	29.76	**154,914.76**	225,741.99	**318.60**	509.13
	8	**559,253,421.11**	4,523,882,220.53	27.08	**21.32**	**155,403.23**	222,249.78	**388.94**	553.40
	9	**477,574,200.07**	4,475,817,547.97	20.52	**14.52**	**162,949.16**	234,863.90	**449.84**	632.71
	10	**453,275,708.29**	5,006,511,747.65	**14.18**	16.31	**168,079.73**	260,435.77	**455.41**	736.25
Pr1002	2	**13,197,140,379.29**	13,363,646,950.99	**0.71**	1.41	**315,511.95**	326,981.21	**105.90**	109.42
	3	**10,548,533,829.38**	15,029,639,532.45	**31.90**	52.60	**311,687.93**	381,549.91	**101.75**	230.82
	4	**6,515,194,464.68**	12,660,835,480.27	55.07	**49.54**	**306,613.98**	384,091.64	**169.03**	315.34
	5	**4,533,183,261.54**	11,563,025,270.34	23.37	**13.03**	**316,234.34**	394,060.44	**160.73**	398.73
	6	**3,455,665,662.18**	11,629,345,061.77	29.96	**29.10**	**315,869.25**	399,895.76	**192.93**	433.53
	7	**3,130,636,186.51**	13,495,565,905.21	**24.68**	37.04	**332,828.43**	447,142.04	321.79	**283.73**
	8	**2,900,682,782.35**	15,636,334,982.12	**16.30**	28.96	**334,429.44**	481,279.29	**320.17**	654.06
	9	**2,490,198,787.51**	14,951,536,387.86	12.40	**6.65**	**339,741.17**	508,850.61	**392.52**	739.29
	10	**2,317,942,833.19**	11,863,806,927.16	**20.65**	36.62	**349,174.72**	511,895.16	**457.86**	839.99

Best values marked in bold.

Table A-1. Performance evaluation results

Article's contents are provided on an Attribution-Non Commercial 4.0 Creative commons International License. Readers are allowed to copy, distribute and communicate article's contents, provided the author's and Journal of Industrial Engineering and Management's names are included. It must not be used for commercial purposes. To see the complete license contents, please visit https://creativecommons.org/licenses/by-nc/4.0/.