Unsupervised Fuzzy Clustering-based Genetic Algorithms to Traveling Salesman Problem

Khalid Jebari[1,*], Abdelaziz El moujahid[1], Abdelaziz Bouroumi[1,2] and Aziz Ettouhami[1]
[1]Laboratoire Conception et Systèmes (Microélectronique et Informatique), Faculty of Sciences,
Mohammed V–Agdal University, UM5A,
Rabat, Morocco

[2] Modeling and Instrumentation Laboratory, Ben M'sik Faculty of Sciences,
Hassan II Mohammedia-Casablanca University, UH2MC
Casablanca, Morocco

***Abstract*—In this paper a new genetic algorithm based on an unsupervised fuzzy clustering is proposed for Traveling Salesman Problem (TSP). The proposed algorithm involves on three phases. In the first phase, the cities are divided into several sub-tours by the clustering algorithm. In the second phase, each partition of cities is considered as a smaller scale TSP problem and this smaller size TSP problem is solved by a genetic algorithm which gets an optimal sub-tour of the cities of this partition. In the third phase, a new technique for connecting all these sub-tours into an appropriate tour of whole cities. As well as, this appropriate tour is improved by a genetic algorithm for cluster centers and a heuristic method. The computer simulations on some standard test problems show good performance for the proposed algorithm.**

Keywords-component; Genetic Algorithms; Unsupervised Learning; Fuzzy Clustering; Traveling Salesman Problem.

I. Introduction

The traveling salesman problem (TSP) is a NP-complete problem [1] where the computation cost will increase rapidly with the increasing of the size of the problem. Given n cities $V_1,V_2,V_3,...,V_n$ and a distance matrix $D = [d_{ij}]$, where d_{ij} is the distance between V_i and V_j, TSP requires finding a tour through all the cities, visiting each cities only once, and returning to the originating one, in order that the total traveled distance is minimized [2]. More formally the goal is to find a permutation P of the cities that minimizes the sum of the distances:

$$f(P(1),...,P(n)) = \sum_{i=1}^{n-1} d\left(V_{P(i)},V_{P(i+1)}\right) + d\left(V_{P(n)},V_{P(1)}\right) \qquad (1)$$

The TSP has a diverse practical application such as manufacturing, examination timetabling, robotics, Integrated circuit testing.

Several such approximation techniques, typically based on search heuristic have been proposed in the literature, including the local search algorithms [3], simulated annealing [4], tabu search [5] elastic nets [6], neural network [7], genetic algorithms [1, 8-11], ant colonies [6] and particle swarm [3]. Furthermore, other approaches have been obtained by combination of local search and simulated annealing [12], by combination of local search and genetic algorithms [2, 13, 14], genetic algorithms and ant colony systems [15].

In this paper, Unsupervised Fuzzy Clustering Genetic Algorithm (UFCGA), which is a novel genetic algorithm based on an unsupervised fuzzy clustering algorithm is proposed for TSP. UFCGA consists of three phases. In the first phase, the cities are divided into several groups by the clustering algorithm. Clustering of the cities makes the calculations much simpler. The search space for the solution increases as the number of cities decreases and vice-versa. If there are N cities then the search space will be N!, and the computational time gets higher accordingly[16]. In order to reduce the mathematical complexity, N value should be reduced and this is achieved by clustering. In the second phase, each group of cities are considered as a smaller scale TSP problem and this smaller size TSP problem is solved by a genetic algorithm which get an optimal sub-tour. In the third phase, a connection scheme is proposed to connect these sub-tours into an appropriate tour of all cities. Besides, this appropriate tour is improved by genetic algorithm and heuristic search scheme. The rest of the paper is organized as follows: section II gives a brief description of clustering algorithms used for division of cities, section III is dedicated to the presentation of a GA used for each cluster. Section IV presents the connection of clusters. Section V discusses the numerical results. Finally, section VI contains our concluding remarks.

II. Clustering Algorithm

In the absence of information about the distribution of cities, the fuzzy clustering offers a better possibility of modeling and management of overlapping clusters, so we opt for a fuzzy classification. Thus our approach is to introduce a fuzzy clustering based on three phases. The first one a unsupervised fuzzy learning (UFL)[17] phase that starts by generating the first class is around the first city encountered. Then a new cluster created when the current city presents a small similarity, less than a prefixed threshold S_{min} , to the entire already existing cluster centers.

UFL() is described as follows:

```
// Vi The City i ; c: Number of Clusters
// d(V_i,V_j) Euclidean distance;
// $C_i$: Cluster Center i (Prototype)
//n:number of cities
-Choose a similarity threshold $S_{min}$
// Initialization
c=1
Cluster Center for the first Class $C_1=V_1$
For (i=2;i<=n;i++)
{
```

$$S(i,j)=\left(1-\frac{d^2(V_i,V_j)}{2}\right) \tag{2}$$

```
  IfS(i,j)<$S_{min}$ {
     c++;
     $C_i=V_i$;
  }
  Else
     Update $C_j$  //j=1,...,c;
```

$$C_j=\frac{\sum_{k=1}^{n}U_{k,j}^{m}V_j}{\sum_{k=1}^{n}U_{k,j}^{m}} \tag{3}$$

```
\\ $m \in [1,+\infty[$
```

Figure 1. Pseudo code of UFL

The second phase is an optimization procedure that ameliorates the learned partition, generated during the previous phase. It uses the fuzzy c-means algorithms (FCM) [18]. This clustering is sensitive of the choice of the data similarity, so criterion validity is uses in third phase. More formally, the proposed clustering algorithm can be described by the following pseudo-code:

```
$m_{min}$=1.1;
$m_{ma}$x=3.1;
$m_{pa}$s=0.1;
pas =0.01;
c*= 2;   //c*:Number of clusters for minimal entropy
$h_{min}$ =1;  //$h_{min}$ : Minimal entropy
$S_{min}$=0.1
$S_{max}$=0.99
For (m = $m_{min}$; m <= $m_{max}$; m = m + $m_{pas}$)
{
   For (seuil=$S_{min}$; seuil<$S_{max}$; seuil=seuil+pas)
      //$S_{max}$ : Maximum similarity
      {
         Apply UFL to The cities tour // Figure 1
         Apply FCM to The cities tour
         Calculate h () using Eq(4)
```

$$h(U)=-\frac{1}{\log(c)}\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{c}[u_{ij}\log(u_{ij})]. \tag{4}$$

```
//h() Entropy
         // U: matrix of membership degree
         // c: Number of clusters
         If ($h_{min}$>h () ) {
            $h_{min}$=h();
            c* = c;
            C*=C;
            U*=U
         }
      }
}
Use  U*;
Use C*;
```

Figure 2. Pseudo code of UFL-FCM-VAL

III. Genetic Algorithm

The number of clusters is given by UFL-FCM-VAL(), then we try to find optimal sub-tour for cities of each cluster by GA.

In order to conceive a genetic solution to TSP we have to determine the encoding method. Then the fitness function in eq 1 is used for assessing and comparing the shortest traveling tour. Thus, starting from an initial population of randomly generated individuals (tour). We evolved this population toward better solutions according to the rules of selection strategy, crossover and mutation. The details are as follows.

The ordinal encoding [11] scheme was used in the UFCGA. Under this scheme, for a cluster, the set of vertices of cluster tour is denoted as T_i. If cardinal (card) of Ti is s, card (T_i) =s, these vertices are denoted as 1, 2, · · ·, s, respectively for representing each cluster cities. Then chromosomes which represent the traveling paths are permutation $j_1, j_2, \cdots, j_s$ of 1, 2, · · · , s.

The initialization of the population of chromosomes is generated by a random process. Each chromosome is represented as permutation $j_1, j_2, \cdots, j_s$ of 1, 2, · · · , s. The process does not yield any illegal chromosome.

We used the Partially-Matched Operator [12]. For Two Parents P_1, P_2

- Pick crossover points A and B randomly $1 \le A \le B \le S$
- Copy the cities between A and B from P_1 into the child.
- For parts of the child's array outside the range [A, B], copy only those cities from P_2, which haven't already been taken from P_1.
- Finally, to fill in the gaps, use the cities that have not yet been taken.

We used the Swap Mutation [19].

1. Choose two points A,B randomly $1 \le A \le B \le S$
2. Swap the genes at position A,B

We used a modified Tournament Selection, which guards in each iteration the best individual. The following pseudo code summarizes the Selection Method.

```
/ /N: population size
$T_{\_alea}$: array of integer containing the indices
```

```
of individuals (Tour) in the population
T_ind_Winner : an array of individuals indices 's
who will be selected
L_sorted : a list of all individual indices sorted
 in decreasing fitness values
l = 0
k=0
For (i=0; i<k; i++)
{
        Shuffle T_alea ;
        For (j=0; j<N; j=j+k+1)
        {
                C1 = T_alea(j);
                For (m=1; m<k; m++)
                {
                        C2=T_alea(j+m);
                        if f(C1)< f(C2) C1 = C2

      // f(Ci): Fitness of individual Ci
                }
                T_ind_Winner(l) = C1
              T_ind_Winner(l+1) = L_sorted (l)
                l=l+2;
              k=k+1
        }
}
```

Figure 3. Tournament Selection Modified

Suppose that the Clustering Algorithm gives, for example, U clusters for TSP.

In the following the GA for cities in one cluster I ($I \in [1, U]$) is given:

```
1. The population sizes N1, · · · ,NU for U clusters
    (T1, .... ,TU)
    pc: crossover probability
    pm: mutation probability
2. Randomly generate a population P(0) contain Ni individuals (Tours)
   (i.e., Ni permutations of 1,2 ,..., Si) Si number of cities for Cluster Ti
   For the i-th cluster, i = 1, 2, ..... , U.
3. Evaluate the individuals for Cluster Ti
4. Condition1= $\frac{Number\ \ of\ \ the\ best}{N} * 100 < 95$
  tmax=100 // Maximum of generations ; t=0
5. While (condition1 AND t<tmax){
   t=t+1;
4. Select P (t) From P (t-1) using Tournament
   Selection Modified in Figure 3
5. Apply PMX crossover with probability pc
6. Apply   Swap mutation with probability pm
7. Create new population P'(t)
8. P (t) = Generational Replacement (P'(t),P(t))
  // Proportion de P'(t) and P (t)
9. Evaluate P (t)                    }
10. Return the best Tour (Individual)
```

Figure 4. GA for a Cluster

IV. CONNECTION OF CLUSTERS

After an optimal sub-tour for each cluster obtained by Algorithm in figure 4 it is necessary to connect these sub-tours to form an optimal tour for the all cities.

Let that Clustering Algorithms generated U clusters which $C_1, \cdots, C_u$ are the cluster centers.

Note, that in this paper, we consider the two-dimensional (2-D) Euclidean TSP, in which the cities lie in R^2, so the coordinates of their centers can be calculated by formulas (3) and are denoted by: $(x_1,y_1),\ldots\ldots(x_u,y_u)$.

The objective of using GA is to form an entire tour that is as short as possible based on the tours generated in the previews section. In each generation, the aim is to evolve candidate solutions, which are permutations of the tour formed by the cluster centers $C_1, \ldots, C_U$. The details for GA are as follow.

The ordinal encoding scheme was used in the UFCGA.
The initialization of the population of chromosomes is generated by the following heuristic:

```
1. Choice vertex C0 randomly;
2. Insert C0 randomly at position
  [E ((U+1/2)-3),E((U+1/2)+3)]
  //U: number of Cluster Centers;
3. The nearest vertex C1 from C0 is selected and inserted
   in front of C0, the second nearest vertex C2
   from C0 is selected and inserted behind C0;
4. The nearest vertex C3 from C1 is selected and inserted
   In front of C1, the nearest vertex C4 from C2 is
   selected and inserted behind C2;
5. Iterate 4 while building the first initial candidates solutions;
6. Repeat (1 to 5) for building the first population P(0).
```

Figure 5. Heuristic Initialization

We used the Partially-Matched Crossover.

We used the Inversion Mutation.

The inversion mutation [20] is randomly select a sub tour, removes it from the tour and inserts it in a randomly selected position. However, the sub tour is inserted in reversed order.

We used a modified Tournament Selection, such as in figure 3.

The GA is similar to that of Figure 4 with modifications at the initialization and mutation operator.

```
1. The population of Cluster Centers size N
     pc: crossover probability
     pm: mutation probability
2. Initialization Use Algorithm in figure 5 P(0)
3. Evaluate the individuals for Cluster Centers
4. condition1= $\frac{Number\ \ of\ \ the\ best}{N} * 100 < 95$
  tmax=100 // Maximum of generations ; t=0
```

```
5.While (condition1 AND t<$t_{max}$){
   t=t+1;
4. Select  P(t) From P(t-1) using Tournament Selection
Modified in Figure 3
5. Apply PMX crossover with probability $p_c$
6. Apply   Inversion mutation with  probability $p_m$
7. Create new population P'(t)
8. P(t)= Generational Replacement (P'(t),P(t)) // Proportion de
P'(t) and P(t)
9. Evaluate P(t)                    }
10. Return the best Tour (Individual)
```

Figure 6. GA for Cluster Centers

The connection algorithm is as follows.

```
Suppose that $T_{O(1)},...,T_{O(u)}$ is the optimal tour given
by the GA in figure 6, note that O(i) is an ordinal
function, which characterizes the order of the tour.

For(i=1;i<U; i++){
// U: Number of  Clusters given by the
// Clustering  Algorithms.
1.Select an edge A1B1 from $T_{O(i)}$ nearest $T_{O(i+1)}$
  Select an edge A2B2 from $T_{O(i+1)}$ nearest  $T_{O(i)}$
2. Calculate :
  d(A1,X)+d(B1,Y) – (d(A1,B1)+d(A2,B2))=min (
d(A1,X)+d(B1,Y) – (d(A1,B1)+d(A2,B2)))
  with  $x,y$ □$\{A2,B2\} X \neq Y$
3.where X = A2 and Y = B2 or X = B2 and Y = A2.
  Suppose that X = A2 and Y = B2.
  Connect A1 and A2 to form an edge A1A2, and
  connect B1 and B2 to form another edge B1B2.
  Delete edges A1B1 and A2B2.
  Then the two nearest clusters are unified to form
  a new cluster.
                                              }
```

Figure 7. Connection Scheme

V. Unsupervised Fuzzy Clustering

Based on previous sections, a new genetic algorithm can be given as follows.

```
1. Use UFL-FCM-VAL Algorithm (figure 2) to generate a
number of clusters (suppose this number is U) of n cities
2. Use Algorithm in figure 4 for each cluster
   (1,...,U) to get the optimal sub-tour for each cluster.
3. Use Algorithm in figure 6 to get the optimal
    tour for cluster centers.
4. Use Algorithm in figure 7 to connect all
   sub-tours for all clusters, to get an appropriate tour
   for all cities.
```

Figure 8. UFCGA

VI. Experimental Results

In this section we present the experimental results of the proposed algorithm. The UFCGA presented above is computer coded in C++ on a 2,0 GHz Core i7 personal computer and tested with a benchmark problem set:

- lin105, a 105-city problem that has an optimal distance value 14379,
- a280, a 280-city problem that has an optimal distance value 2579,
- lin318, a 318-city problem that has an optimal distance value 42029,
- att532, a 532-city problem that has an optimal distance value 27686,
- rat783 a 738-city problem that has an optimal distance value 8806,
- pr1002, a 1002-city problem that has an optimal distance value 259045,
- nrw1379, a 1379-city problem that has an optimal distance value 56638,
- u2319, a 2319-city problem that has an optimal distance value 234256,
- pcb3038, a 3038-city problem that has an optimal distance value 137694,
- rL5915, a 5915-city problem that has an optimal distance value 565530,
- pla7397, a 7397-city problem that has an optimal distance value 23260728.

These problems are available from [21].

TABLE I. The Relative Error in Percentage for UFCGA and SGA

Problem	Technique Used		
	UFCGA		***SGA***
	Number of Clusters	***REP***	***REP***
Lin105	5	0.00	1.15
A280	5	0.00	5.53
Lin318	7	0.03	9.05
Att532	10	0.09	11.97
Rat738	10	0.12	19.35
pr1002	15	1.16	23.54
nrw1379	15	1.31	29.72
u2319	18	2.12	36.15
pcb3038	20	2.9	39.23
rL5915	20	5.4	41.07
pla7397	20	9.3	43.89

We adopt the following parameter values: population size (N=500; 1000; 2000). Crossover and mutation probabilities are $p_c = 0.8$, and $p_m = 0.02$, respectively.

We compared the results found by UFCGA and those found by a standard GA (SGA). SGA used the following parameters:

We used the Partially-Matched Crossover, the Inversion Mutation and Tournament Selection with different tournament size (2; 4; 8).

The initialization of the population of chromosomes is generated by a random process.

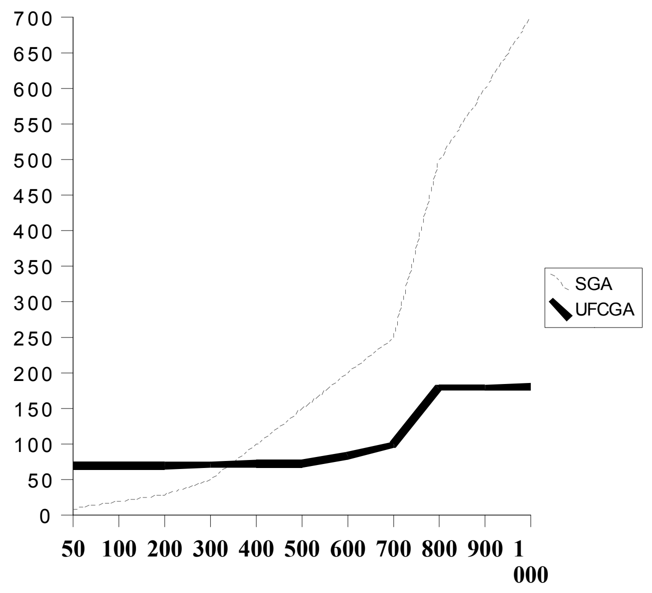

Figure 9. Computing time for UFCGA and SGA
(X axis: Number of Cities, Y Axis: Computing Time in Seconds)

Table I provides sample results related to an aspect of this study. It is the aspect of quality assessment of the optimum provided by the SGA and UFCGA. To measure this quality, we used the Relative Error in Percentage (REP)

$$\frac{\Delta f}{f} = \frac{\left| f^{*} - f \right|}{f} \qquad (5)$$

Where f^{*} the optimum provided by the algorithm and f the actual optimum, which is a priori known [21]. The cities number varies from 100 to 7397, i.e. medium and large cities numbers. The results demonstrate clearly the efficiency of the algorithm. Note that the first two tests the optimum was found in 20 generations. Note also, that the running time of the algorithm was reasonable.

In table I, the column labeled by “Number of Clusters” was especially for UFCGA not for SGA.

Figure 9 shows clearly that the computing time (CT) of SGA is higher than UFCGA. The CT is almost linear for UFCGA, whereas for SGA, the CT increases considerably.

VII. Conclusion

In this paper we deal with TSP problem with large and medium number of cities. Our basic idea is to classify cities, with a Fuzzy unsupervised learning algorithm and FCM, into several partitions, in order to reduce the burden of mathematical complexity. After the clustering phase we applied the GA for each cluster of cities. To make the connection between the different clusters of cities, first we applied again the GA for the cluster centers and secondly, we used a heuristic algorithm to join the edges of clusters whose cluster centers are the optimal tour given by GA.

UFCGA has the ability of finding optimal solutions with small amount of computation.

The simulation results demonstrate the effectiveness of the proposed algorithm.

The UFCGA shows superior performance compared to the traditional genetic algorithms. For medium size problems, we obtained optimal solutions. On a larger size problem, UFCGA generated best solutions. Thus, we may conclude that the UFCGA is efficient methodology for the TSP with large number of cities.

In future work. We will consider implementing UFGA in a parallel architecture, for investigating the efficiency of the parallel computation.

References

[1] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning. Boston, MA, USA: Addison-Wesley Longman, 1989.

[2] K. Katayama and H. Narihisa, "Iterated local search approach using genetic transformation to the traveling salesman problem," 1999, pp. 321-328.

[3] F. Greco, "Travelling Salesman Problem," I-Tech Education and Publishing KG, Croatia, 2008.

[4] G. Gutin and A. P. Punnen, The traveling salesman problem and its variations vol. 12: Kluwer Academic Pub, 2002.

[5] R. Durbin, et al., "An analysis of the elastic net approach to the traveling salesman problem," Neural Computation, vol. 1, pp. 348-358, 1989.

[6] M. Dorigo and L. M. Gambardella, "Ant colonies for the travelling salesman problem," BioSystems, vol. 43, pp. 73-82, 1997.

[7] L. I. Burke and J. P. Ignizio, "Neural networks and operations research: An overview," Computers & operations research, vol. 19, pp. 179-189, 1992.

[8] T. Bäck, Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms: Oxford University Press, USA, 1996.

[9] K. Bryant, "Genetic Algorithms and the Traveling Salesman Problem," Department of Mathematics, Harvey Mudd College, 2000.

[10] P. Larranaga, et al., "Genetic algorithms for the travelling salesman problem: A review of representations and operators," Artificial Intelligence Review, vol. 13, pp. 129-170, 1999.

[11] Y. Wang, et al., "A new encoding based genetic algorithm for the traveling salesman problem," Engineering Optimization, vol. 38, pp. 1-13, 2006.

[12] H. Kargupta, et al., "Ordering genetic algorithms and deception," Urbana, vol. 51, p. 61801, 1992.

[13] B. Freisleben and P. Merz, "New genetic local search operators for the traveling salesman problem," in Parallel Problem Solving from Nature PPSN IV, 1996, pp. 890-899.

[14] P. Merz and B. Freisleben, "Genetic local search for the TSP: New results," in Evolutionary Computation, 1997., IEEE International Conference on, 1997, pp. 159-164.

[15] S. M. Chen and C. Y. Chien, "Parallelized genetic ant colony systems for solving the traveling salesman problem," Expert Systems with Applications, vol. 38, pp. 3873-3883, 2011.

[16] G. Reinelt, The traveling salesman: computational solutions for TSP applications vol. 840: Springer-Verlag, 1994.

[17] A. Bouroumi, et al., "Unsupervised Fuzzy Learning and Cluster Seeking," Intelligent Data Analysis, vol. 4, pp. 241-253, September 2000.

[18] J. C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms. New York: Plenum Press, 1981.

[19] D. B. Fogel, "An evolutionary approach to the traveling salesman problem," Biological Cybernetics, vol. 60, pp. 139-144, 1988.

[20] D. B. Fogel, "Applying evolutionary programming to selected traveling salesman problems," Cybernetics and systems, vol. 24, pp. 27-36, 1993.

[21] http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp.