International Journal of Soft Computing 6 (3): 68-74, 2011
ISSN: 1816-9503
© Medwell Journals, 2011

A Parallel Genetic Approach for the TSP

[1]Khalid Jebari, [1]Abdelaziz Bouroumi and [2]Aziz Ettouhami
[1]Laboratoire Conception et Systmes (Microlectronique et Informatique),
Faculty of Sciences, Mohammed V-Agdal University UM5A, Rabat, Morocco
[2]Modeling and Simulation Laboratory, Ben Msik Faculty of Sciences,
Hassan II Mohammedia-Casablanca University, UH2MC, Casablanca, Morocco

Abstract: The study deals with the efficiency of the parallel computation of the Travelling Salesman Problem (TSP) using the genetic algorithms and an unsupervised fuzzy clustering. First, the cities are classified by a clustering algorithms. Second, each class of cities is considered as a sub-tour TSP problem. A parallel genetic algorithms is used for solving the sub tour TSP problem. The main aim by creating the parallel algorithm is to accelerate the execution time of solving TSP. A connection method is proposed to connect the sub-tours into a global tour of whole cities. Furthermore, this global tour is resolved by genetic algorithms for cluster centers and a heuristic scheme. Experimental results, on Master Slave architecture with different TSP problemes show the efficacy of the proposed algorithm in parallelism exploitation.

Key words: Parallel algorithms, genetic algorithms, unsupervised learning, fuzzy clustering, traveling salesman problem, Parallel Virtual Machine (PVM)

INTRODUCTION

TSP is a NP hard problem (Greco, 2008) where the computation cost will increase with it size. Lets V_1, V_2, V_3, ..., V_n, n cities and a distance matrix D = $[d_{ij}]$ where d_{ij} is the distance between V_i and V_j, TSP requires to find the order in which the salesman visit each city just once and return to the first city as to minimize the total distance traveled (Gutin and Punnen, 2002; Laporte and Palekar, 2002; Reinelt, 1994). More formally the aim is to find a permutation π of the cities that minimizes the sum of the distances:

$$f\left(\pi(1),\ldots,\pi(n)\right)=\sum_{i=1}^{n-1} d\left(V_{\pi(i)},V_{\pi(i+1)}\right)+\left(V_{\pi(n)},V_{\pi(1)}\right) \quad (1)$$

For finding solution to TSP, several approximation techniques have been proposed in the literature. The researchers find local search algorithms (Johnson, 1990), simulated annealing (Van Laarhoven and Aarts, 1988), tabu search (Fiechter, 1994) elastic nets (Durbin *et al*., 1989), neural network (Burke and Ignizio, 1992), genetic algorithms (Back, 1996; Bryant, 2000; Fogel, 1988; Larranaga *et al*., 1999; Wang *et al*., 2006) ant colonies (Dorigo and Gambardella, 1997) and particle swarm (Greco, 2008). In addition, other approaches have been obtained by combination of local search and simulated annealing, (Martin and Otto, 1996) by combination of local search and genetic algorithms (Freisleben and Merz, 1996; Katayama and Narihisa, 1999; Lin and Kernighan, 1973; Merz and Freisleben, 1997), genetic algorithms and ant colony systems (Chen and Chien, 2010).

In this study a novel Parallel Genetic Algorithm (PGA) (Alba, 2005) based on a Unsupervised Fuzzy Clustering algorithm is proposed for TSP called Parallel Hybrid Genetic Algorithm (PHGA). PHGA consists of two phases. First, the cities are classified by a clustering algorithm. Clustering of the cities makes the calculations much reduced. If there are N cities, the search space is N! then, the computational time will be higher (Reinelt, 1994). 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 sub tour TSP problem and this sub tour TSP problem is solved by a parallel genetic algorithm witch get an optimal sub-tour.

For the parallelization, the well known Master Slave (Nedjah *et al*., 2006) method was used. In this way, each C (C: Number of clusters given by unsupervised fuzzy clustering) slaves is associated to a sub tour TSP and performs it while the master connects these sub-tours into a feasible tour of whole cities. This feasible tour is improved by genetic algorithm and heuristic search scheme. PHGA algorithm is the same as the sequential

Corresponding Author: Abdelaziz Bouroumi, Laboratoire Conception et Systmes (Microlectronique et Informatique), Faculty of Sciences, Mohammed V-Agdal University UM5a, Rabat, Morocco

one but the time consuming is reduced. Also the researchers have used C slaves to speed up the execution of a sequential algorithm.

CLUSTERING ALGORITHMS

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, the researchers have opted for a fuzzy classification. Thus, the approach is to introduce a fuzzy clustering based on three phases. The first one is the Unsupervised Fuzzy Learning (UFL) (Bouroumi *et al*., 2000) phase that starts by generating the first class which 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. The second phase is the Fuzzy C-Means algorithms (FCM) (Bezdek, 1981), its purpose is to ameliorate the learned partition generated during the previous phase. This clustering is sensitive of the choice of the data similarity so, criterion validity is used in the third phase. More formally, the proposed clustering algorithm UFL_FCM_VAL() is described as follows:

Pseudo code of UFL-FCM-VAL:

```
UFL_FCM_VAL()
{
m_min = 1.1;
 m_max = 3.1;
 m_pas = 0.1;
 pas = 0.01;
 c* = 2
//c* :Number of clusters for a 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 for The cities tour //
         Apply FCM for The cities tour
         Calculate h () //Eq. 2
```

$$h(U) = -\frac{1}{\log(c)}\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{c}[u_{ij}\log(u_{ij})] \qquad (2)$$

```
//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*;
}
```

Pseudo code of UFL:

```
UFL
{
//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++)
   {
```

$$S(i,j) = \left(1 - \frac{d^2(V_i, V_j)}{2}\right) // \qquad (3)$$

```
If S(i,j)<S_min {
                c++;
                C_i = V_i;
              }
Else
      Update C_j // j = 1,...,c ;
```

$$C_j = \frac{\sum_{k=1}^{n} U_{k,j}^{m} V_j}{\sum_{k=1}^{n} U_{k,j}^{m}} // \qquad (4)$$

```
m ∈ [1,∞] , the researchers choose m = 1;
  }

 // When V_j is replaced in Eq. 3 by C_i the relation
// is interpreted as membership degree of V_i
// to the jth Cluster:
```

$$U_{i,j} = \left(1 - \frac{d(V_i, C_j)}{\sqrt{2}}\right) // \qquad (5)$$

```
Update U;
Update C;
}
```

GENETIC ALGORITHMS

After the researchers have got a number of clusters by the cluster algorithm the researchers then, try to find optimal sub-tour for cities of each cluster by GA.

In order to conceive a genetic (Back, 1996; Goldberg, 1989) solution to TSP, the researchers have to determine the Encoding method. Then, the fitness function in Eq. 1 is used for assessing and comparing the shortest traveling tour. Hence, starting from an initial population of randomly generated individuals (tour), the researchers evolved this population toward better solutions according to the rules of the selection strategy, crossover and mutation.

Encoding method (Kargupta *et al*., 1992; Wang *et al*., 2006): The ordinal encoding scheme was used in the PHGA. 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 $j_1, j_2, \ldots, j_s$ of 1, 2, ... , s.

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

Crossover operator: The researchers have used the Partially-Matched Operator (Larranaga *et al*., 1999). For Two parents P_1, P_2:

- Pick crossover points A and B randomly $1 \leq A \leq B \leq S$
- 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

Mutation operator: The researchers have used the Swap Mutation (Fogel, 1993):

- Choose two points A, B randomly $1 \leq A \leq B \leq S$
- Swap the genes at position A, B

Selection operator: The researchers have used a modified tournament selection which guards in each iteration the best individual. The following pseudo code summarizes the selection method.

Tournament selection modified:

```
TOSM{
// 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
 indecreasing 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 (k)
            l = l+2;
          k = k+1;
      }
}
}
```

Genetic algorithm for one cluster: The Clustering algorithms give for example, U clusters for TSP. In the following the GA for cities in one cluster is given.

GA for a cluster:

```
GA_for_a_cluster{
The population sizes N1, _E _E _E ,NU for U clusters
( T1, _E _E _E ,TU)
pc: crossover probability
pm: mutation probability
Randomly generate a population P(0) contain Ni individuals (Tours)
(i.e., Ni permutations of 12 _E _E _E Si) Si number of cities for Cluster Ti
for the ith cluster, i = 1, 2, _E _E _E , U.
Evaluate the individuals for Cluster Ti
```

$$\text{condition1} = \frac{\text{Number of the best}}{N} * 100 < 95$$

```
tmax = 100 // Maximum of generations ; t = 0
While (condition1 AND t<tmax){
t = t+1;
Select P(t) From P(t-1) using Tournament
Selection Modified in tournament selection modified
Apply PMX crossover with probability pc
Apply Swap mutation with probability pm
Create new population P'(t)
P(t) =  Generational Remplacement (P'(t),P(t))
// Proportion de P'(t) and P(t)
Evaluate P(t) }
Return the best Tour (Individual)
}
```

CONNECTION OF CLUSTERS

After an optimal sub-tour for each cluster obtained by algorithm, it is necessary to connect these sub-tours to form an optimal tour for the whole cities. Suppose that clustering algorithms generated U clusters which $C_1, \ldots, C_u$ are the cluster centers.

In this study, the researchers consider the two Dimensional (2D) Euclidean TSP in which the cities lie in R^2 so, the coordinates of their centers can be calculated by Eq. 4 and are denoted by: $(x_1,y_1),\ldots.. (x_u,y_u)$.

The objective of using GA is to form an entire tour that is as short as possible based on the tours generated in the previous 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 follows.

Encoding method: The ordinal encoding scheme has been used in the PHGA.

Initialization: The initialization of the population of chromosomes has been generated randomly.

Mutation operator: The researchers have used the inversion mutation. The inversion mutation (Fogel, 1993) randomly selects a sub tour, removes it from the tour and inserts it in a randomly selected position. However, the sub tour is inserted in a reversed order.

Selection operator: The researchers have used a modified tournament selection.

Genetic algorithm for cluster centers: The GA is similar to that of GA for cluster with modifications at the initialization and mutation operator.

GA for cluster centers:

GA_cluster(){
The population of Cluster Centers size N
p_c: crossover probability
p_m: mutation probability
Initialization Use Algorithm in figure 5 P(0)
Evaluate the individuals for Cluster Centers

$$\text{condition1}=\frac{\text{Number of the best}}{N}*100 < 95$$

t_{max}= 100 // Maximum of generations ; t = 0
While (condition1 AND t<t_{max}){
t = t+1;
Select P(t) From P(t-1) using Tournament Selection Modified
Apply PMX crossover with probability p_c
Apply Inversion mutation with probability p_m
Create new population P'(t)
P(t) = Generational Remplacement (P'(t),P(t)) // Proportion de P'(t) and P(t)
Evaluate P(t) }
Return the best Tour (Individual)
}

Connection scheme:

Connection_scheme(){
Suppose that $T_{O(1)},...,T_{O(u)}$ is the optimal tour given
by the GA, 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.
Select an edge A1B1 from $T_{O(i)}$ nearest $T_{O(i+1)}$
Select an edge A2B2 from $T_{O(i+1)}$ nearest $T_{O(i)}$
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 $x,y \in \{A2,B2\}\ X \neq Y$
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.
}
}

PARALLEL HYBRID GENETIC ALGORITHMS

Based on previous sections, a PHGA can be given as follows (Fig. 1):

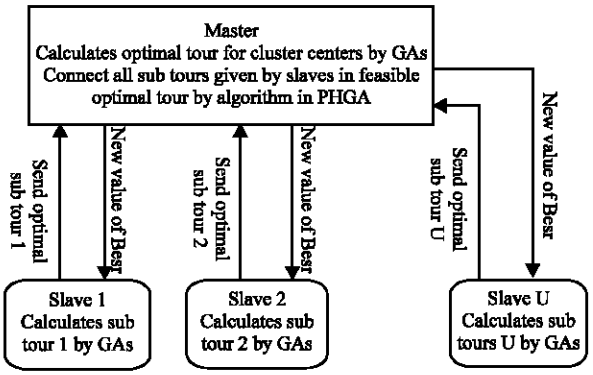

Fig. 1: Master slave architecture for PHGA

- The classication algorithm give a partition of cities into a certain number of clusters (U clusters). The researchers associate each cluster to a slave
- When the master in master-slave architecture, calculates the optimal tour for the cluster centers by genetic algorithms, each slave (U slaves) calculates in parallel the optimal sub tours by Gas also
- The calculation of the overall optimal tour by connections scheme algorithms is done only after a certain number of times of generations
- Steps 2 and 3 are repeated until the number of generation is >1000 or a satisfaction solution. The master send two parameters: (Best, Tmax) for performing the result in each next generation

Pseudo code of PHGA:

PHGA (){
Use Algorithm UFL-FCM-VAL() to generate a number of clusters
//(suppose This number is U) of n cities
Tid = pvm_mytid();
best tour length = MAX INT;
Best = 60; tmax = 100;
generations = 0; count = 10;
M_S[0] = pvm_parent();
for(i = 1; i<U; i++)
pvm_spawn (, , , , , &M_S[i]);
Repeat {
generations = generations + 1;
if (M_S[0]< 0) { // master;
M_S[0]< =Tid
GA cluster()
if ((generations % count) = 0) { for (all slaves)
pvm_recv(optimal_sub_tour)
optimized tour lengt = Connection Scheme();
// Algorithm Connection Scheme() to connect all sub-tours
// to get a feasible tour for all cities ;
}
best tour length = optimized tour length;
} else{ //slave;
pvm_initsend();
GA for a Cluster() //Use in Parallel GA for a Cluster() for each slave (1,...,U)
//to get the optimal sub-tour ;
pvm_send(optimal tour);
pvm_recv(Best, tmax);
} if (Best< 96) {

```
Best = Best+10;
tmax = tmax+300; // For Performing the Optimal tour

}

}until (generations > tmax)or () Number of the best / N * 100 < Best
}
pvm_exit();
```

EXPERIMENTAL RESULTS

In this study the researchers present the experimental results of the proposed algorithm. The PHGA presented above is computer coded in C++. The researchers have considered clusters of Pentium 4, 3.6 GHz and 1 GB DDR RAM running GNU/Linux Debian 5.0 as Operating System they are connected through a via Fast Ethernet switch (100 Mbps) and the software environments was Parallel Virtual Machine (PVM) is based on the PVM3 (Geist *et al*., 1994). PHGA is 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 (http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/).

The researchers have adopted 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.

The researchers have compared the results found by PHGA and those found by a standard GAs (SGAs). SGAs have used the following parameters:

Crossover operator: The researchers have used the partially-matched crossover.

Table 1: The Relative error in percentage for PHGA and SGA

	Technique used		
	PHGA		SGA
Problems	Number of slaves	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	2.90	39.23
rL5915	20	5.40	41.07
pla7397	20	9.30	43.89

Table 2: Speedup for PHGA

Problem	Speedup
Lin105	49.8
A280	48.5
Lin318	39.2
Att532	31.8
Rat738	21.1
pr1002	18.8
nrw1379	17.6
u2319	13.5
pcb3038	11.2
rL5915	8.5
pla7397	6.8

Mutation pperator: The researchers have used the inversion mutation.

Selection operator: The researchers have used a Tournament Selection with different tournament size (Back, 1996; Bezdek, 1981; Chen and Chien, 2010).

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

Table 1 shows the sample results related to an aspect of this study. It is the aspect of quality assessment of the optimum provided by the SGA and PHGA. To measure this quality, the researchers used the Relative Error in Percentage (REP):

$$\frac{\Delta f}{f} = \frac{(f^* - f)}{f} \quad (6)$$

Where:

f^* = The optimum provided by the algorithm
f = The actual optimum which is a priori known (Nedjah *et al*., 2006)

The cities number varies from 100-7397 i.e., medium and large cities numbers. The results demonstrate clearly the efficiency of the algorithm. In the first two tests, the optimum was found in 20 generations. Note also that the running time of the algorithm was reasonable.

In Table 1, the column labeled by Number of Clusters was especially for PHGA not for SGA.

Table 2 shows sample results related to another aspect of this study. It is the aspect of the speedup. This

study considers the conventional de_nition of the parallel speedup of an algorithm as the ration of the execution time of the best serial algorithms Ts and the execution time of the parallel program, T_p:

$$S_p = \frac{T_s}{T_p} \quad (7)$$

By analysing this Table 2 the researchers can see clearly that the proposed method performs better than the standard genetic algorithms.

CONCLUSION

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

The PHGA has the ability of finding optimal solutions with small amount of computation. The simulation results demonstrate the effectiveness of the proposed algorithm. The PHGA shows superior performance compared to the standard genetic algorithms. For medium size problems, the reseachers have obtained optimal solutions. On a larger size problem, PHGA generated best solutions and the execution time is shorter than SGAs. Thus,the reseachers may conclude that the PHGA is an efficient methodology for the TSP with large number of cities.

In future research, the reseachers will consider implementing PHGA in an another parallel architecture, to investigate and compare the efficiency of the parallel computation.

REFERENCES

Alba, E., 2005. Parallel Metaheuristics: A New Class of Algorithms. John Wiley and Sons, New Jersey, USA., ISBN-13: 9780471678069, pp: 554.

Back, T., 1996. Evolutionary Algorithms in Theory and Practice. 1st Edn., Oxford University Press, New York, USA., ISBN-13: 978-0195099713, pp: 328.

Bezdek, J.C., 1981. Pattern Recognition with Fuzzy Objective Function Algorithms. 1st Edn., Kluwer Academic Publishers, Norwell, MA, USA.

Bouroumi, A., M. Limouri and A. Essaid, 2000. Unsupervised fuzzy learning and cluster seeking. Intell. Data Anal., 4: 241-253.

Bryant, K., 2000. Genetic algorithms and the traveling salesman problem. Ph.D. Thesis, Department of Mathematics, Harvey Mudd College, Claremont, California, USA.

Burke, I.L. and J.P. Ignizio, 1992. Neural networks and operations research: An overview. Comput. Oper. Res., 19: 179-189.

Chen, S.M. and C.Y. Chien, 2010. Parallelized genetic ant colony systems for solving the traveling salesman problem. Expert Syst. Appl., 38: 3873-3883.

Dorigo, M. and L.M. Gambardella, 1997. Ant colonies for the travelling salesman problem. Biol. Syst., 43: 73-81.

Durbin, R., R. Szeliski and A. Yuille, 1989. An analysis of the elastic net approach to the traveling salesman problem. Neural Comput., 1: 348-358.

Fiechter, C.N., 1994. A parallel tabu search algorithm for large traveling salesman problems. Discrete Applied Math., 51: 243-267.

Fogel, D.B., 1988. An evolutionary approach to the traveling salesman problem. Biol. Cybern., 60: 139-144.

Fogel, D.B., 1993. Applying evolutionary programming to selected traveling salesman problems. Int. J. Cybernet. Syst., 24: 27-36.

Freisleben, B. and P. Merz, 1996. New genetic local search operators for the traveling salesman problem. Proceedings of the 4th Conference on Parallel Problem Solving fron Nature, (PPSN'96), Springer-Verlag, Berlin, Germany, pp: 890-899.

Geist, A., A. Beguelin, J. Dongarra, W. Jiang, R. Manchek and V.S. Sunderam, 1994. PVM: Parallel Virtual Machine: A Users Guide and Tutorial for Networked Parallel Computing. MIT Press, Massachusetts, Cambridge, ISBN: 0-262-57108-0.

Goldberg, D.E., 1989. Genetic Algorithms in Search Optimization and Machine Learning. Addison Wesley Publishing Co., Reading, Massachutes.

Greco, F., 2008. Travelling Salesman Problem. I-Tech Education and Publishing, Croatia, pp: 75-115.

Gutin, G. and A.P. Punnen, 2002. The Traveling Salesman Problem and its Variations. 1st Edn., Kluwer Academic, Dordrecht, ISBN: 1-4020-0664-0.

Johnson, D.S., 1990. Local optimization and the traveling salesman problem. Proc. Colloquium Automata, Languages Program., 433: 446-461.

Kargupta, H., K. Deb and D.E. Goldberg, 1992. Ordering Genetic Algorithms and Deception. In: Parallel Problem Solving from Nature, Manner, R. and B. Manderick (Eds.). Elsevier Science Publishers B.V., Berlin, Germany, pp: 47-56.

Katayama, K. and H. Narihisa, 1999. Iterated local search approach using genetic transformation to the traveling salesman problem. Proceedings of the Genetic and Evolutionary Computation Conference, July 13-17, Morgan Kaufmann, New York, USA., pp: 321-328.

Laporte, G. and U. Palekar, 2002. Some applications of the clustered travelling salesman problem. J. Operat. Res. Soc., 53: 972-976.

Larranaga, P., C. Kuijpers, R. Murga, I. Inza and S. Dizdarevic, 1999. Genetic algorithms for the travelling salesman problem: A review of representations and operators. Artif. Intell. Rev., 13: 129-170.

Lin, S. and B.W. Kernighan, 1973. An effective heuristic algorithm for the traveling-salesman problem. Operat. Res., 21: 495-516.

Martin, O.C. and S.W. Otto, 1996. Combining Simulated annealing with local search heuristics. Ann. Operat. Res., 63: 57-75.

Merz, P. and B. Freisleben, 1997. Genetic local search for the TSP: New results. Proceedings of the IEEE International Conference on Evolutionary Computation, April 13-16, IEEE Xplore Press, Indianapolis, USA., pp: 159-164.

Nedjah, N., E. Alba and L.D.M. Mourelle, 2006. Parallel Evolutionary Computations. Springer-Verlag, Berlin Heidelberg, ISBN-13: 9783540328377, pp: 200.

Reinelt, G., 1994. The Traveling Salesman: Computational Solutions for TSP Applications. Springer-Verlag, Berling, Germany, ISBN-13: 9783540583349, pp: 223.

Van Laarhoven, P.I.M. and E.H.L. Aarts 1988. Simulated Annealing: Theory and Application. Kluwer Academic Publishers, Boston, New York, ISBN: 9789027725134.

Wang, Y., L. Han, L. Yinghua and S. Zhao, 2006. A new encoding based genetic algorithm for the traveling salesman problem. Eng. Optim., 38: 1-13.