A Parallel Genetic Approach for the TSP

Khalid Jebari, Abdelaziz Bouroumi and Aziz Ettouhami

Laboratoire Conception et Systèmes (Microélectronique et Informatique), Faculty of Sciences, Mohammed V-Agdal University UM5A, Rabat, Morocco
Modeling and Simulation Laboratory, Ben M'sik 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 problems 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 \( V_1, V_2, V_3, \ldots, V_n \) 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; Reimelt, 1994). More formally the aim is to find a permutation \( \pi \) of the cities that minimizes the sum of the distances:

\[
I(\pi(1), \ldots, \pi(n)) = \sum_{i=1}^{n-1} d(V_{\pi(i)}, V_{\pi(i+1)}) + d(V_{\pi(n)}, V_{\pi(1)})
\]  

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 (Friechter, 1994) elastic nets (Durbin et al., 1989), neural network (Burke and Ignizio, 1992), genetic algorithms (Back, 1996; Bryant, 2000; Fogel, 1988; Larrañaga 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 Narishia, 1999; Lin and Kernighan, 1973; Merz and Freisleben, 1997), genetic algorithms and ant colony systems (Chen and Chen, 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 (Reimelt, 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 which 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 Systèmes (Microélectronique 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 $S_{max}$ 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$_{step} = 0.1$;
pas = 0.01;
c$_{star} = 2$;
    //c$_{star}$ : Number of clusters for a minimal entropy
    h$_{min} = 1$;
    //h$_{min}$ : Minimal entropy
    $S_{max} = 0.1$
    $S_{min} = 0.99$
    For (m = m$_{min}$; m <= m$_{max}$; m = m + m$_{step}$) {
        For (i = 1; i <= m; i++) {
            $c_i$ = V$_i$
            Get $U^*_i$ from $S(j)$ and $S_i$
            Update $c_i$

        }
    }

    h(U) = \frac{1}{ \log(c_n)} \sum_{i=1}^{n} \sum_{j=1}^{c_n} [u_{ij} \log(u_{ij})]

    //h() Entropy
    // U: matrix of membership degree
    // c: Number of clusters
    If $h_{min} < h()$ {
        $h_{min} = h()$
        c$_{star} = c$
        $U^* = U$
    }
}

Use $U^*$;
Use c$_{star}$;

**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 $T_i$ 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 $j_1, j_2, ..., j_s$ of 1, 2, ..., $s$. The process does not yield any illegal chromosome.

**Initialization:** The initialization of the population of chromosomes is generated by a random process. Each chromosome is represented as permutation $j_1, j_2, ..., j_s$ 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_j: array of integer containing the indices of individuals (Tour) in the population
T_j,swap: an array of individuals indices's who will be selected
L_note: a list of all individual indices sorted in increasing fitness values
i = 0
k = 0
For (i = 0; i < N; i++)
    Shuffle T_j,swap:
    For (j = 0; j < N; j = j + k + 1)
        C_i = T_j,swap(j);
        For (m = 1; m < k; m++)
            C_i = T_j,swap(j + m);
        if ($R(C_i) < R(C_2)$) $C_1 = C_2$

// $R(C_i)$: Fitness of individual $C_i$

**GA for a cluster:**

GA_for_a_cluster{

The population sizes $N_E, E, E, E, N_u$ for U clusters
($T_{u, E, E, E, E, E, E}$)
$p_c$ : crossover probability
$p_m$ : mutation probability
Randomly generate a population $P(0)$ contain $N_E$ individuals (Tours)
(i.e., N_E permutations of 12 E E E E E S) number of cities for Cluster $T_i$
for the $i$th cluster, $i = 1, 2, ..., E, E, E, E, E, U$.
Evaluate the individuals for Cluster $T_i$

condition 1: $\text{Number of the best} \leq 100 < 95$

$t_{\max } = 100$ // Maximum of generations : t = 0
While (condition1 AND $t < t_{\max }$){

    Select $P(t)$ from $P(t-1)$ using Tournament
    Selection Modified in tournament selection modified
    Apply PMX crossover with probability $p_c$
    Apply Swap mutation with probability $p_m$
    Create new population $P(t)$
    $P(0) = \text{Generational Replacement} (P(0), P(t))$
    // Proportion de $P(t)$ and $P(t)$
    Evaluate $P(t)$

Return the best Tour (Individual)

}
**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:**

\[
\text{GA\_cluster}() \{
\begin{align*}
&\text{The population of Cluster Centers size } N \\
&p_c; \text{crossover probability} \\
&p_m; \text{mutation probability} \\
&\text{Initialization Use Algorithm in figure 5 P}(0) \\
&\text{Evaluate the individuals for Cluster Centers} \\
&\text{condition1}\begin{cases}
&\text{Number of the best } 100 < 95 \\
&\text{condition2}\end{cases}
\text{N}
\begin{cases}
&\text{for } t = 0 \\
&\text{t} = t + 1; \\
&\text{Select } P(0) \text{ From } P(t-1) \text{ using Tournament Selection Modified} \\
&\text{Apply PMX crossover with probability } p_c \\
&\text{Apply Inversion mutation with probability } p_m \\
&\text{Create new population } P(t) \\
&\text{P}(t) = \text{Generational Replacement}(P(0), P(t)) \text{ / Proportion de } P(0) \text{ and } P(t) \\
&\text{Evaluate } P(t) \}
&\text{Return the best Tour (Individual)}
\end{cases}
\]

**Connection scheme:**

Suppose that \( T_{c0}, \ldots, T_{cU} \) is the optimal tour given by the GA, note that \( O() \) is an ordinal function which characterizes the order of the tour.

For \( i = 1;1; U)\{
\begin{align*}
&\text{U; Number of Clusters given by the} \\
&\text{// Clustering Algorithms.} \\
&\text{Select an edge } A1B1 \text{ from } T_{c_u} \text{ nearest } T_{c_{(i-1)}} \\
&\text{Select an edge } A2B2 \text{ from } T_{c_{(i+1)}} \text{ nearest } T_{c_{(i)}} \\
&\text{Calculate : } \\
&d(A1,X)+d(B1,Y)-d(A1,B1)+d(A2,B2) = \\
&\text{min} (d(A1,X)+d(B1,Y) - d(A1,B1)+d(A2,B2)) \\
&\text{with } x,y \in \{A2,B2\} X \neq Y \\
&\text{where } X = A2 \text{ and } Y = B2 \text{ or } X = B2 \text{ and } Y = A2 \\
&\text{Suppose that } X = A2 \text{ and } Y = B2 \\
&\text{Connect A1 and A2 to form an edge A1A2, and} \\
&\text{connect B1 and B2 to form another edge B1B2} \\
&\text{Delete edges A1B1 and A2B2} \\
&\text{Then the two nearest clusters are unified to form} \\
&\text{a new cluster.}
\end{align*}
\}

**PARALLEL HYBRID GENETIC ALGORITHMS**

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

- The classification architecture 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:**

```pseudo
PHGA ()
Use Algorithm UFL-PCM-VAI() to generate a number of clusters
// (suppose This number is U) of n cities
Tid = pmv\_mytid();
best tour length = MAX INT;
Best = 60; Tmax = 100;
generations = 0; count = 10;
M_S[i] = pmv\_parent();
for(i = 1; i < U; i++)
    pmv\_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)
        pmv\_recv(optimal sub tour)
        optimized tour length = 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,
        pmv\_intense() ;
        GA for a Cluster() // Use in Parallel GA for a Cluster() for each slave (1...U)
        // to get the optimal sub-tour
        pmv\_send(optimal tour);
        pmv\_recv(Best, Tmax);
    } if (Best < 90) {
```

Fig. 1: Master slave architecture for PHGA
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 IV, 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
- pbl3038, a 3038-city problem that has an optimal distance value 137694
- l5915, 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.

Mutation operator: The researchers have used the inversion mutation.

Selection operator: The researchers have used a Tournament Selection with different tournament size (Back, 1996; Bezdok, 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):

\[
\Delta f = \frac{(f - f^*)}{f^*}
\]  

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

<table>
<thead>
<tr>
<th>Problems</th>
<th>Number of clusters</th>
<th>PHGA</th>
<th>SGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lin105</td>
<td>5</td>
<td>0.00</td>
<td>1.15</td>
</tr>
<tr>
<td>A280</td>
<td>5</td>
<td>0.00</td>
<td>5.53</td>
</tr>
<tr>
<td>Lin318</td>
<td>7</td>
<td>0.03</td>
<td>9.03</td>
</tr>
<tr>
<td>Att532</td>
<td>10</td>
<td>0.09</td>
<td>11.97</td>
</tr>
<tr>
<td>Rat783</td>
<td>10</td>
<td>0.12</td>
<td>19.35</td>
</tr>
<tr>
<td>pr1002</td>
<td>15</td>
<td>1.16</td>
<td>23.54</td>
</tr>
<tr>
<td>nrw1379</td>
<td>25</td>
<td>1.31</td>
<td>29.72</td>
</tr>
<tr>
<td>u2319</td>
<td>18</td>
<td>2.12</td>
<td>36.15</td>
</tr>
<tr>
<td>pbl3038</td>
<td>20</td>
<td>2.90</td>
<td>39.23</td>
</tr>
<tr>
<td>l5915</td>
<td>20</td>
<td>5.40</td>
<td>41.07</td>
</tr>
<tr>
<td>pla7397</td>
<td>20</td>
<td>9.30</td>
<td>43.89</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problems</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lin105</td>
<td>48.8</td>
</tr>
<tr>
<td>A280</td>
<td>48.5</td>
</tr>
<tr>
<td>Lin318</td>
<td>39.2</td>
</tr>
<tr>
<td>Att532</td>
<td>31.8</td>
</tr>
<tr>
<td>Rat783</td>
<td>21.1</td>
</tr>
<tr>
<td>pr1002</td>
<td>18.8</td>
</tr>
<tr>
<td>nrw1379</td>
<td>17.6</td>
</tr>
<tr>
<td>u2319</td>
<td>13.5</td>
</tr>
<tr>
<td>pbl3038</td>
<td>11.2</td>
</tr>
<tr>
<td>l5915</td>
<td>8.5</td>
</tr>
<tr>
<td>pla7397</td>
<td>6.8</td>
</tr>
</tbody>
</table>
study considers the conventional definition of the parallel speedup of an algorithm as the ratio of the execution time of the best serial algorithms $T_s$ and the execution time of the parallel program, $T_p$.

$$S_p = \frac{T_s}{T_p}$$  \hfill (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 researchers 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 researchers applied the GA for each cluster of cities in parallel architecture. To make the connection between the different clusters of cities, first the researchers have applied again the GA for the cluster centers and secondly, the researchers 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 researchers have obtained optimal solutions. On a larger size problem, PHGA generated best solutions and the execution time is shorter than SGAs. Thus, the researchers may conclude that the PHGA is an efficient methodology for the TSP with large number of cities.

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

REFERENCES


