A Parallel Ring Method for Solving a Large-scale Traveling Salesman Problem

Roman Bazylevych
Lviv Polytechnic National University, Lviv, Ukraine
University of Information Technology and Management, Rzeszow, Poland
E-mail: rbaz@polynet.lviv.ua

Bohdan Kuz and Roman Kutelmakh
Lviv Polytechnic National University, Lviv, Ukraine
Email: bohdankuz@gmail.com, rkutelmakh@ua.fm

Rémy Dupas
a Univ. Bordeaux, IMS, UMR 5218, F-33405 Talence, France
b CNRS, IMS, UMR 5218, F-33405 Talence, France
E-mail: remy.dupas@gmail.com

Bhanu Prasad*
Department of Computer and Information Sciences, Florida A&M University, Tallahassee, Florida, USA
E-mail: bhanu.prasad@famu.edu, *Corresponding author

Yll Haxhimusa
Section of Artificial Intelligence, Medical University of Vienna, Austria and Faculty of Electrical and Computer Engineering, University of Prishtina, Kosova
E-mail: yll.haxhimusa@meduniwien.ac.at

Lubov Bazylevych
Institute of Applied Problems of Mathematics and Mechanics NASU, Lviv, Ukraine
E-mail: lbaz@iapmm.lviv.ua

Abstract—A parallel approach for solving a large-scale Traveling Salesman Problem (TSP) is presented. The problem is solved in four stages by using the following sequence of procedures: decomposing the input set of points into two or more clusters, solving the TSP for each of these clusters to generate partial solutions, merging the partial solutions to create a complete initial solution $M_0$, and finally optimizing this solution. Lin-Kernighan-Helsgaun (LKH) algorithm is used to generate the partial solutions. The main goal of this research is to achieve speedup and good quality solutions by using parallel calculations. A clustering algorithm produces a set of small TSP problems that can be executed in parallel to generate partial solutions. Such solutions are merged to form a solution, $M_0$, by applying the “Ring” method. A few optimization algorithms were proposed to improve the quality of $M_0$ to generate a final solution $M_f$. The loss of quality of the solution by using the developed approach is negligible when compared to the existing best-known solutions but there is a significant improvement in the runtime with the developed approach. The minimum number of processors that are required to achieve the maximum speedup is equal to the number of clusters that are created.

Index Terms—TSP, parallelization, combinatorial optimization, clustering.

I. INTRODUCTION

Traveling Salesman Problem (TSP) is NP-hard therefore, for large-scale problems, it is challenging to produce optimal solutions in a reasonable time. Many problems in integrated circuit manufacturing, scheduling, analysis and synthesis of chemical structures, logistic problems, robotics, etc. can be modeled and solved as TSP problems. A lot of research has been done on using various techniques to generate good quality solutions that are close to the optimal [14, 15, 16, 20, 22, 39, 40, 43].

Local heuristic approaches are one of the main approaches to obtain near optimal tours for large instances of TSP in a short time [28, 39]. Both discrete optimization [28, 29] as well as continuous optimization [30, 31, 32] methods are used to solve the Euclidean as well as non-Euclidean TSP.

Many existing heuristic solutions for solving TSP with $n$ points have $O(n^2)$ or higher complexity hence they are not efficient for solving large-scale TSPs. Lin-Kernighan...
heuristic is based on “tour-improvement” – it accepts a
given tour and modifies it to obtain an alternative tour of
lesser cost [20]. Chained Lin-Kernighan heuristic
improves an existing solution and it overcomes some
drawbacks in local optimization algorithms [2].
Applegate et al. [1] have calculated the optimal solution
for a 85,900-points TSP - the largest problem that is
solved optimally until now. This experiment required
nearly 136 years of CPU time. Well-known test-cases for
large-scale TSP problems contain up to 10^7 points [24]
but, until now, there are no optimal solutions found for
such problems. Interesting results were achieved in the
challenge usa115475 problem, started by Cook, that has
115,475 points [25]. This large-scale problem started by
Cook was solved independently by Clarist, Helsgaun, and
Nagata [26]. Although the tour length of the solutions
presented by these three researchers is exactly the same
and is equal to 6,204,999, the three tours are mutually
different. This problem inspired us to find optimal
solutions for such large-scale problems. Using the
heuristic presented in this research, we generated a
solution with length 6,205,118, which is 0.0019% worse
than the solution presented independently by Clarist,
Helsgaun, and Nagata.

There have been a number of studies of human
performance on the TSP [33, 34, 35, 36, 37, 38]. It is
quite certain that human subjects do not search the whole
or even substantial parts of the problem space when
solving the combinatorial optimization problems because
of memory and cognitive processing limitations, in spite
of massive parallel processing of brain. This means that
human subject might be using a kind of abstraction of
data in solving the TSP. Many researchers have shown
that humans produce close-to-optimal solutions to TSP in
a time that is (on average) proportional to the number of
cities [34, 35]. Note that the TSP instances were small
(less than 150 cities) when experimenting with human
subjects. Motivated by the human vision system, more
specifically the vision fovea, the authors in [14, 22, 35,
38] used hierarchical representation (pyramid) to achieve
abstraction of the data (cities). The main processes in
obtaining a hierarchical model are clustering methods
using merging or division principles [41]. For example,
the cities are merged into bigger and bigger modes (called
clusters), then an initial tour solution on the modes is
produced [14]. A splitting strategy could be followed as
well. The whole data space is split over and over again
into cells of cities, creating modes [22]. Afterward, a
refinement on this initial solution is done on modes,
mimicking a human fovea, until the tour contains all the
cities, producing an approximate solution for the TSP.
This strategy is in fact a divide-and-conquer strategy,
which seems to be plausible, taking the limiting
processing power and space limitation of human brain.

Some TSP algorithms can be executed in parallel by
decomposing the given set of points into clusters [4, 5, 6,
7, 8]. Bazylevych et al. [7] have proposed a “Ring”
method that merges the TSP solutions obtained from
different clusters into a complete initial solution. The goal
of the current research is to reduce the computing time by
using a large number of parallel processors. The
maximum speedup was achieved when the number of
processors is equal to the number of clusters into which
the given set of points is divided. The current research is
an extension to the work of Bazylevych et al. [8] that
used a single processor. In the current research, the
experiments are conducted on a distributed system that
consists of many processors (a maximum of 64
processors) running in parallel.

The rest of the paper is organized as follows. Section 2
focuses on the problem formulation. Section 3 describes
the parallel ring method, including the decomposition of
given points into clusters, solving the TSP for each such
cluster to generate partial solutions, merging the partial
solutions to generate a complete initial solution, and
optimizing the complete initial solution. Implementation
and experimental results are presented in Sections 4 and 5
respectively. Finally, conclusions are presented in Section
6.

II. PROBLEM FORMULATION

Given a set \( P = \{p_1, p_2, ..., p_n\} \) of \( n \) points that are
described by their coordinates \( p_i = (x_i, y_i), \ i = 1 \ to \ n \) and
the Euclidean distance function dist: \( P \times P \rightarrow \mathbb{R} \) that
defines the distance between any two points then the
problem is to find out a closed route \( M \), with minimal
length \( L \), that starts from a point and visits each and every
point only once and returns back to the starting point.

If \( \text{dist}(p_i, p_j) = \text{dist}(p_j, p_i) \) then this problem is
symmetric else it is asymmetric. The problem considered
in this research is of symmetric type. The problem is a
kind of Hamilton cycle problem – finding a cycle in an
undirected graph that visits each vertex exactly once and
have a minimal length. The main goal of the current
research is to develop a methodology to find a high
quality shortest solution by using a large number of
parallel processors.

III. THE PARALLEL RING METHOD

A. Main Stages

The ring method presented in this research is used to
solve TSPs with a very large number of points. Currently,
the best approaches for solving TSP are Concorde [9] and
LKH heuristics [16]. Non parallel and non distributed
algorithms can be applied directly for small-scale TSPs.
Distributed and decomposition approaches to solve TSPs
were described in [12, 13, 17, 18, 19, 23].

The developed parallel approach for solving a large-
scale TSP consists of the following stages:

1. Decomposing the set, \( P \), of given points into
clusters along with constrains on the number of
points in each cluster.
2. Finding the partial TSP solutions for each of these
clusters independently and in parallel.
3. Merging the partial solutions to get a complete
initial solution $M_0$.

4. Optimizing $M_0$ to get the final solution $M_f$.

The input set of points, $P$, is divided into two or more clusters (subsets) with each of these clusters contain a limited number of points. Several decomposition algorithms are investigated to divide $P$ into clusters [6, 7, 8]. The maximum number of points in each cluster is $n_{\text{max}}$. The decomposition of $P$ into different clusters can be done very fast without using parallel algorithms. As shown in the later sections of this research (see Table 1), if $P$ contains $10^6$ points then it requires around 15 seconds time for the decomposition using sequential algorithms. That is why we did not develop parallel algorithms for decomposition, although it is possible. Great reduction in the overall computational time is achieved during the second and the third stages i.e., while finding the partial TSP solutions for each of the clusters and while merging them. To find the partial TSP solutions for each of the clusters, we used LKH software [16]. The result of the second stage is the partial TSP solutions for each of the clusters. The sum of the length of all these partial TSP solutions is represented as $M_{2\text{ps}}$.

The runtime of the complete initial solution $M_0$, (i.e., the runtime of the TSP solution for the entire $P$) of the developed approach is equal to $T = t_c + t_p + t_o$, where $t_c$ is the time to divide the set $P$ into clusters, $t_p$ is the time required to find all the partial TSP solutions and $t_o$ is the time required to merge these partial solutions. The maximum degree of parallelism (i.e., the maximum number of processors that can be used in parallel) for finding the partial TSP solutions is equal to the total number of clusters: however, in this research, the maximum number of processors is not used: only a maximum of 64 processors are used in parallel. The duration of finding the partial TSP solutions in parallel is $t_p \approx t_o$, where $t_o$ is the time to solve the TSP in a single cluster that has $n_{\text{max}}$ points. The partial solutions are merged to form a complete initial solution $M_0$. The merging process uses adjacent clusters. All the clusters together form a planar graph, if the border between any two adjacent clusters is considered as an edge and the ends of those edges are considered as vertices of the planar graph. Three or more different borders meet at every vertex. The chromatic number of that planar graph is 4, according to the four color theorem [3]. This number determines the number of steps needed to find, in parallel, the complete initial solution at the third stage. More details on this matter are provided in Section 3C.

The number of number of vertices of the planar graph, when each group painted in a different color, shows the maximum degree of parallelism achieved while merging the partial solutions. If the number of points in the ring is same as that of a cluster then the time for finding the TSP solution in one ring is approximately equal to the time for finding a partial TSP solution for the cluster. More precisely, the time for finding the TSP solution in one ring is little bit more than that of finding a partial TSP solution for the cluster because there are additional operations such as finding the temporary route segments (see Section 3C for more details). The time required to perform these additional operations is not much when compared to the time for finding the TSP solution within the ring.

In general, $t_o > 4 \cdot t_0$. This is because a pair of rings may overlap even if they are not adjacent (see Section 3C for more details) hence such rings must be considered sequentially but not in parallel to find their TSP solutions. In that case, the number of steps required to join the clusters (i.e., Stage 3 of the developed approach) will be more than 4, where 4 is the chromatic number of the created planar graph.

Based on the experimental results, $t_c << t_0$. For example, for Mona Lisa test-case [21] with 100,000 points, $t_c \approx 1.5$ seconds and $t_0 = 956$ seconds. The runtime for the complete solution $T > t_c + t_o + 4 \cdot t_0$. More precisely, $T > 5 \cdot t_0$.

The maximum speedup for solving the entire TSP was achieved when the number of available processors is equal to the total number of clusters formed. Some procedures for optimization of the complete solution (Stage 4) can also be performed in parallel. The main restriction for the optimization is that the local optimization areas (Section 3D contains more details about the local optimization areas), where parallel optimization is performed, should not overlap with each other and different optimization algorithms must be applied successively.

The speedup, $g$, of the developed approach is:

$$g = \frac{T(n)}{T(\frac{n}{k})} = \frac{T(n)}{\frac{5 \cdot T(n_0)}{5 \cdot T(n/k)}} = \frac{T(n)}{5 \cdot T(n/k)},$$

where $T(n)$ is the runtime to solve the entire TSP without decomposition, $k$ is the total number of clusters formed, $n$ is the total number of points in $P$, and $T(n_0) = T(n_{\text{max}})$ is the runtime to solve TSP in a cluster that contains $n_{\text{max}} = n / k$ points.

Consider a TSP algorithm that has $k$ clusters and has time complexity $O(m^n)$, where $m$ is a nonnegative integer, for a data set with $n$ points. The speedup $g$ that can be achieved by the developed decomposition approach, when compared to the considered TSP algorithm is:

$$g < \frac{n^m}{5 \binom{n}{k}^m} = \frac{k^m}{5}.$$ 

More precisely, $g < \frac{k^m}{5}$.

The minimum number of processors required to achieve the maximum speedup is determined by the number, $k$, of clusters that are created. Because $n_{\text{max}} << n$ and $k = n / n_{\text{max}}$, a significant gain in the runtime is achieved while obtaining the complete solution in parallel.

B. Decomposition

During the first stage of TSP solution process, the input set of points, $P$, is divided into clusters. The
Delaunay triangulation [10, 42] is applied on $P$. The result is a set of triangular faces. By having Delaunay triangulation, no element of $P$ will be inside the circumcircle of any triangle. The total length of Delaunay triangulation edges is minimal. Only the edges of the Delaunay triangulation are considered while finding the solution (i.e., while dividing the points into clusters). The algorithm for clustering consists of the following main steps: take one triangle from the center of the surface and consider its points as a first fragment of the first cluster; create the new cluster around the current fragment by adding new triangles in every next front (step) of the equal wave propagation from the current fragment by triangles in Delaunay triangulation that are adjacent to the triangles of previous front or by successively adding new triangles having the smallest area in each step; stop building the current cluster if the number of points in it is “close” to the maximum number $n_{\text{max}}$; take one of the adjacent triangles of the already created clusters and consider it as the first fragment of new cluster and create if there are more free triangles available; recursively repeat this operation; end the clustering process if there are no more free triangles available; search for all small clusters that have less than $n_{\text{min}}$ points, merge such two adjacent clusters together, or with a neighboring smallest cluster even if the merged cluster has more than $n_{\text{max}}$ points.

In Fig. 1, the set $P$ has 160,000 points, the number of clusters is 156, the number of clusters after merging the smallest clusters is 135, approximate number of points in each cluster is 1000, and the lowest number of points in a cluster is 800. Pareja160K (TSP Art [24]) is shown in Fig. 1(a). Its decomposition by using equal wave propagation from the current fragment is shown in Fig. 1(b). The result by successively adding new triangles (having the smallest area) in every step is shown in Fig. 1(c).

C. Finding and merging the partial solutions

In the second stage, separate TSP solutions are found for each cluster. These solutions are merged together in the third stage to create an initial solution $M_s$ for the whole problem. Fig. 2 contains the partial TSP solutions for some pieces of the Van Gogh test-case. In the second stage, each partial solution can be found in parallel, as the calculations for finding the partial solutions are independent of each other.

The ring method [7, 8] is used to merge the partial solutions into one complete initial solution $M_s$. The process consists of the following steps (Fig. 3):

1. Build an initial ring: Randomly select a cluster. The border of this cluster is created by the points that belong to the border triangles’ edges of this cluster (see Fig. 4). Every border edge has two incident triangles, one is internal and the other one is external to the cluster. The points of these triangles create an initial ring. These triangles are considered as the first front (step) for wave propagation. Every next front of wave propagation is performed by adding new points, from the adjacent triangles, to the current ring’s fragment.

2. Build the full ring: It consists of all the points that are obtained by wave propagation from the initial ring to a given number rin of internal triangles and a given number rout of external triangles. If the number rin is large enough, the final ring covers the entire cluster. The number of steps that needs to be performed for wave propagation is a parameter of the algorithm and this parameter depends on the number of points that must belong (approximately) to the ring. The ring creates overlapping zone that consists some points of the cluster that is currently considered (could be full but not necessarily) and some points of all the adjacent clusters. The number of wave propagation steps that needs to be performed in the internal and external cluster zones of the ring could be different.
3. Replace all the pieces of routes that belong to the ring’s outside zones (external and internal) by temporary, fixed, and single edges having zero length. Let us call these temporary, fixed, and single edges as “temporary route segments”.

4. Solve the TSP in the ring, by considering all its points in the ring and the temporary route segments (formed in Step 3) by applying the chosen TSP algorithm.

5. Replace the temporary route segments by the real route pieces that have been removed in Step 3.

6. Repeat all the previous steps for all other clusters that are not yet considered.

To improve the initial solution, \( M_0 \), the entire surface (all points of set \( P \)) should be covered by such LOAs and the adjacent LOAs must overlap with each other. The quality of the final solution depends on the following parameters of the algorithm: the size (number of points) \( n_{out} \) in the LOA and the size of the overlapping areas \( n_{oa} \) between LOAs. Increasing these sizes will improve the quality of the solution but it will increase the runtime. Some tradeoff must be made between these sizes. We proposed several methods to cover the entire surface by LOAs to improve the quality of the initial solution:

1. SCB - by scanning along the cluster’s border with LOAs (Fig. 5). The centers \( c_i \) and \( c_{i+1} \) of the LOAs are at the cluster’s border points. The points in the LOA are determined by wave propagation around its center with triangles. The border areas of all clusters must be considered.

![Fig.5. Scanning by LOA along the cluster border (SCB) – one step.](image)

2. SR - by scanning along the route (i.e., overall solution) for LOAs (Fig. 6). The centers \( c_i \) and \( c_{i+1} \) of LOAs are on the points of the route. The points in a LOA are determined by wave propagation around its center.

![Fig.6. Scanning by LOA along the route (SR) – one step.](image)

3. SGA - by scanning over the whole surface by given geometrical LOAs such as rectangles (Fig. 7) or other. All the points of every LOA are considered. Some scanning strategy (i.e., the order of sequential steps), for example, horizontal, vertical, zigzag, or spiral, is chosen for this purpose.

For parallel implementation, in Step 1, all the clusters whose rings do not overlap with each other will be considered simultaneously. The number of such steps is more than 4 as the chromatic number of a planar graph is 4. As a result, we receive one complete initial solution \( M_0 \) that passes through all the points of \( P \).

D. Optimization of the solution

The initial solution \( M_0 \) requires further optimization. It can be achieved by reducing the length of the route in the local optimization areas (LOA). The LOA consists of a small number of nearest points where there is a possibility of finding a better TSP (i.e., revising \( M_0 \)). The LOA could be in the form of a circle, rectangle, or another kind of area. If better pieces of the route are found in the LOA then the existing segments of the route are replaced by the corresponding better ones.
different non-adjacent LOAs whose full scanning areas do not overlap with each other. The new solution is accepted for the given area if the route’s length is less than that of the existing solution. The number of subsets of clusters a planar graph that can be considered independently (i.e., to apply the optimization on LOAs in parallel) is not less than \(4\) – the chromatic number of the planar graph whose vertices represent the clusters and the edges join such vertices whose full scanning areas overlap each other. For all clusters in each subset, the TSP can be performed simultaneously.

The experimental results show that the quality of optimization increases with the extension of the scanning and overlapping areas. However, it increases the runtime. An optimal trade-off has to be found. It is suggested to repeat the scanning by changing the parameters and the scanning order (for example, scan in anti-clockwise direction rather than clockwise). Experimental results also show that an improvement is achieved through no more than 3-4 full cycles of optimization by using the same scanning parameters.

Specific critical surface zones (Fig. 8) include 3 or more adjacent clusters. These zones require additional improvements. For optimization in such zones, we use enlarged LOAs with 1200-1500 points or more.

Unfortunately, different optimization strategies cannot be executed in parallel because each optimization strategy requires the whole set of points. However, some procedures in a single optimization strategy that don’t overlap with each other can be executed in parallel.

IV. SOFTWARE FOR THE RING METHOD

Parallel implementation requires efficient communications between the processors. In our experiments, Message Passing Interface was used for communications between the computer nodes. It allows scaling of TSP solver to a distributed system without major changes to the software architecture. The main stages of parallelization using different CPUs are shown in Fig. 9.

One CPU (CPU1) divides the given set of points into \(k\) different clusters. Each cluster is then assigned to a separate CPU. Partial TSP solutions are simultaneously produced from each of the clusters. These partial solutions are merged to form an initial and complete solution \(M_0\). A final optimized solution \(M_f\) is developed from this solution. In the last stage, we use several optimization algorithms to produce a good quality final solution \(M_f\).

V. EXPERIMENTAL RESULTS

The developed parallel ring method was investigated in terms of the quality of the solution and runtime. The TSP test-cases were taken from [11, 24, 25]. Experiments were conducted on a distributed system that contains Intel Xeon E5640 @ 2.67 GHZ CPU processors.
The following parameters were used for merging the partial solutions:

- the number of points in cluster \( n_{cl} = 900; \)
- the internal depth of the ring \( r_{in} = 9 \) triangles (i.e., the ring covered 9 Delaunay triangles when propagating the wave from inside to the given cluster);
- the external depth of the ring \( r_{ext} = 24 \) triangles (i.e., the ring covered 24 Delaunay triangles when propagating the wave from outside to the given cluster).

The relations between different parameters of the clustering process, i.e., triangulation time \( t_t \), number of points in clusters \( n_{cl} \), runtime \( t_c \), and the number of clusters \( k \), are shown in Table 1. Triangulation time \( t_t \) takes approximately 10-15% of the entire decomposition process.

Table 2 shows the dependency of the sum of partial solutions, \( M_{2\text{ps}} \), on the size of the clusters (that are used for decomposition) for Mona-lisa test-case [21]. The table shows that an increment in the number of points in the clusters gives better results but it needs more runtime (the best-known length is 5757191).

Table 3 shows the comparison between the tour lengths before merging (sum of partial solutions \( M_{2ps} \)) and after merging (the initial solution \( M_0 \)) for some tests from TSP Art library [24]. The average quality improvement is approximately 1%. Better results (≈5%) were obtained for test usa115475 because we used larger inside and outside zones.

The dependencies between the duration, speedup, and efficiency (the ratio of speedup to the number of cores) of finding the partial solutions in parallel are shown in Fig. 10 (test E10M.0). The duration and efficiency are going down but speedup is increasing with the number of cores, but not proportionally, because the communication between the cores is increasing.

Table 1. Dependencies in the decomposition process

<table>
<thead>
<tr>
<th>Test-case</th>
<th>Number of points ( n )</th>
<th>Triangulation time (sec)</th>
<th>Number of points ( n_{cl} ) in cluster</th>
<th>Runtime ( t_c ) (sec)</th>
<th>Number of clusters ( k )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mona-L</td>
<td>100000</td>
<td>0.290</td>
<td>1000</td>
<td>1.505</td>
<td>96</td>
</tr>
<tr>
<td>100k</td>
<td>120000</td>
<td>0.357</td>
<td>1000</td>
<td>2.471</td>
<td>48</td>
</tr>
<tr>
<td>Vang.</td>
<td>140000</td>
<td>0.422</td>
<td>1000</td>
<td>2.961</td>
<td>58</td>
</tr>
<tr>
<td>Venus</td>
<td>160000</td>
<td>0.480</td>
<td>1000</td>
<td>3.471</td>
<td>68</td>
</tr>
<tr>
<td>pareja</td>
<td>180000</td>
<td>0.541</td>
<td>1000</td>
<td>3.996</td>
<td>77</td>
</tr>
<tr>
<td>160k</td>
<td>200000</td>
<td>0.602</td>
<td>1000</td>
<td>2.679</td>
<td>177</td>
</tr>
<tr>
<td>courbet</td>
<td>104815</td>
<td>0.198</td>
<td>1000</td>
<td>4.454</td>
<td>88</td>
</tr>
<tr>
<td>238025</td>
<td>238025</td>
<td>0.492</td>
<td>1000</td>
<td>2.351</td>
<td>51</td>
</tr>
<tr>
<td>Ira</td>
<td>498378</td>
<td>1.090</td>
<td>1000</td>
<td>3.131</td>
<td>232</td>
</tr>
<tr>
<td>E1M.0</td>
<td>10^6</td>
<td>3.799</td>
<td>1000</td>
<td>6.820</td>
<td>475</td>
</tr>
<tr>
<td>E10M.0</td>
<td>10^7</td>
<td>74.41</td>
<td>1000</td>
<td>420.9</td>
<td>9906</td>
</tr>
</tbody>
</table>
A Parallel Ring Method for Solving a Large-scale Traveling Salesman Problem

Table 2. Dependency of Partial Solutions Sum $M_{ps}$ from Cluster Sizes for Mona-Lisa Test Case [21]

<table>
<thead>
<tr>
<th>$n_0$</th>
<th>$k$</th>
<th>$t_{ps}$ (sec)</th>
<th>$M_{2ps}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>197</td>
<td>808</td>
<td>5836165</td>
</tr>
<tr>
<td>800</td>
<td>122</td>
<td>3104</td>
<td>5844204</td>
</tr>
<tr>
<td>1100</td>
<td>89</td>
<td>3560</td>
<td>5830572</td>
</tr>
<tr>
<td>1400</td>
<td>69</td>
<td>4408</td>
<td>5816598</td>
</tr>
<tr>
<td>1700</td>
<td>57</td>
<td>4144</td>
<td>5811076</td>
</tr>
</tbody>
</table>

We tried to optimize the solution by scanning along the cluster border several times with the same values for the parameters. The experiment is aimed to find the number of iterations when further optimizations with the same parameters did not give further improvement. Fig. 11 shows the results with the test usa115475 [25] when 900 points were taken in LOAs with a distance of 54 points between them. As we can see, the fifth and further iterations did not give any further improvement in the quality of the solution. Therefore, to improve the quality of the solution further, the best way is to increase the size of LOA and decrease the distance between the LOAs (increasing the overlapping zone) and use different optimization algorithms.

Table 3. Comparison of Tour Lengths before and after Merging for Tests from TSP Art Library [24]

<table>
<thead>
<tr>
<th>Test-case</th>
<th>$M_{ps}$ (sum of partial solutions)</th>
<th>Number of points in cluster $n_0$</th>
<th>Inside depth $r_{in}$</th>
<th>Outside depth $r_{out}$</th>
<th>$M_0$ (initial solution)</th>
<th>% of reducing the ledge after merging</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mona-lisa 100K</td>
<td>5818357</td>
<td>2500</td>
<td>3</td>
<td>6</td>
<td>5763878</td>
<td>0.9452</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>6</td>
<td>5761695</td>
<td>0.9834</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9</td>
<td>6</td>
<td>5760264</td>
<td>1.0085</td>
</tr>
<tr>
<td>Vangogh 120K</td>
<td>6624291</td>
<td>1600</td>
<td>3</td>
<td>6</td>
<td>6551963</td>
<td>1.1039</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>6</td>
<td>6549447</td>
<td>1.1428</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9</td>
<td>6</td>
<td>6548668</td>
<td>1.1548</td>
</tr>
<tr>
<td>Venus 140K</td>
<td>6889593</td>
<td>1600</td>
<td>3</td>
<td>6</td>
<td>6819620</td>
<td>1.0261</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>6</td>
<td>6816208</td>
<td>1.0766</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9</td>
<td>6</td>
<td>6814392</td>
<td>1.1036</td>
</tr>
<tr>
<td>Pareja 160K</td>
<td>7699030</td>
<td>1600</td>
<td>3</td>
<td>6</td>
<td>7630644</td>
<td>0.8962</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>6</td>
<td>7626749</td>
<td>0.9477</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9</td>
<td>6</td>
<td>7624413</td>
<td>0.9787</td>
</tr>
<tr>
<td>Courbet 180K</td>
<td>7989912</td>
<td>1600</td>
<td>3</td>
<td>6</td>
<td>7902880</td>
<td>1.1013</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>6</td>
<td>7899127</td>
<td>1.1493</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9</td>
<td>6</td>
<td>7896762</td>
<td>1.1796</td>
</tr>
<tr>
<td>Earring 200K</td>
<td>8256382</td>
<td>1600</td>
<td>3</td>
<td>6</td>
<td>8184591</td>
<td>0.8771</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>6</td>
<td>8180071</td>
<td>0.9329</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9</td>
<td>6</td>
<td>8177299</td>
<td>0.9671</td>
</tr>
<tr>
<td>Usa 115475</td>
<td>6519843</td>
<td>2000</td>
<td>10</td>
<td>10</td>
<td>6216748</td>
<td>4.8755</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>15</td>
<td>6213891</td>
<td>4.9237</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>20</td>
<td>6210871</td>
<td>4.9747</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>25</td>
<td>6209467</td>
<td>4.9984</td>
</tr>
</tbody>
</table>

Fig.10. Dependencies between duration, speedup, efficiency, and number of cores for finding the partial solutions in parallel for test E10M.0.

To improve the results, we investigated the possibility of sequential use of several optimization algorithms. Optimization by scanning around the cluster borders (algorithm SCB, 9 steps 1-9) was performed during the first phase by taking the merged solution $M_0 = 6294001$ as an initial solution (Fig. 12 and Table 4). Specific critical zones (algorithm SCZ, 8 steps 10-17) were considered for the second phase.

Fig.11. Optimization process with the same parameters for the test usa115475.
Table 4. Two phases of sequential optimization on test-case usa115475

<table>
<thead>
<tr>
<th># of step</th>
<th>(n_{oa})</th>
<th>(n_{sa})</th>
<th>Length</th>
<th>Improvement (%)</th>
<th>Total improvement (%)</th>
<th>Runtime (min)</th>
<th>Total runtime (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHASE 1. Sequential optimization by the scanning along the cluster border (M_s = 6294001)</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>700</td>
<td>38</td>
<td>6235901</td>
<td>0.932</td>
<td>0.932</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>2</td>
<td>725</td>
<td>40</td>
<td>6224553</td>
<td>0.182</td>
<td>1.116</td>
<td>18</td>
<td>37</td>
</tr>
<tr>
<td>3</td>
<td>750</td>
<td>42</td>
<td>6218796</td>
<td>0.093</td>
<td>1.209</td>
<td>17</td>
<td>54</td>
</tr>
<tr>
<td>4</td>
<td>775</td>
<td>44</td>
<td>6215947</td>
<td>0.046</td>
<td>1.256</td>
<td>14</td>
<td>68</td>
</tr>
<tr>
<td>5</td>
<td>800</td>
<td>46</td>
<td>6213997</td>
<td>0.031</td>
<td>1.287</td>
<td>16</td>
<td>84</td>
</tr>
<tr>
<td>6</td>
<td>825</td>
<td>48</td>
<td>6211757</td>
<td>0.036</td>
<td>1.324</td>
<td>17</td>
<td>101</td>
</tr>
<tr>
<td>7</td>
<td>850</td>
<td>50</td>
<td>6210921</td>
<td>0.013</td>
<td>1.338</td>
<td>17</td>
<td>118</td>
</tr>
<tr>
<td>8</td>
<td>875</td>
<td>52</td>
<td>6210578</td>
<td>0.006</td>
<td>1.343</td>
<td>18</td>
<td>136</td>
</tr>
<tr>
<td>9</td>
<td>900</td>
<td>54</td>
<td>6209838</td>
<td>0.012</td>
<td>1.355</td>
<td>20</td>
<td>156</td>
</tr>
<tr>
<td>PHASE 2. Continuation of sequential optimization by considering the critical zones</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>925</td>
<td></td>
<td>6208202</td>
<td>0.026</td>
<td>0.026</td>
<td>42</td>
<td>198</td>
</tr>
<tr>
<td>11</td>
<td>950</td>
<td></td>
<td>6207820</td>
<td>0.006</td>
<td>0.033</td>
<td>44</td>
<td>242</td>
</tr>
<tr>
<td>12</td>
<td>975</td>
<td></td>
<td>6207647</td>
<td>0.003</td>
<td>0.035</td>
<td>43</td>
<td>285</td>
</tr>
<tr>
<td>13</td>
<td>1000</td>
<td></td>
<td>6207549</td>
<td>0.002</td>
<td>0.037</td>
<td>46</td>
<td>331</td>
</tr>
<tr>
<td>14</td>
<td>1025</td>
<td></td>
<td>6207514</td>
<td>0.001</td>
<td>0.037</td>
<td>48</td>
<td>379</td>
</tr>
<tr>
<td>15</td>
<td>1050</td>
<td></td>
<td>6207500</td>
<td>0.000002</td>
<td>0.038</td>
<td>50</td>
<td>429</td>
</tr>
<tr>
<td>16</td>
<td>1075</td>
<td></td>
<td>6207484</td>
<td>0.000003</td>
<td>0.038</td>
<td>52</td>
<td>481</td>
</tr>
<tr>
<td>17</td>
<td>1100</td>
<td></td>
<td>6207446</td>
<td>0.001</td>
<td>0.039</td>
<td>50</td>
<td>531</td>
</tr>
</tbody>
</table>

The experiment shows that it is useful to change the optimization algorithms to improve the quality of the solution. For the first phase, we received 1.4% improvement; and for the second phase, a 0.0001% improvement. Intel Xeon CPU E5-2620 v2 2.10GHz processor was used and parallel calculations were not used in this experiment.

Fig. 12. Optimization process with two algorithms for the test usa115475.

A significant feature of the developed approach is its near linear computational complexity (Table 5 and Fig. 13). Table 5 shows the dependency between the size of the problem, runtime, and the quality without parallelization. Experiments were conducted on a PC with Athlon II X2 240 processor with 2.8 GHz CPU and 2 GB RAM. The following parameters were used:

- The number of points in a cluster \(n_{oa}\) = 800 to 900 and the size of the overlapping area \(n_{sa}\) = 400.
- Internal depth of a ring \(r_{oa}\) = 10 triangles (i.e., the ring covered 10 Delaunay triangles while propagating the wave inside the given cluster). External depth of a ring is \(r_{sa}\) = 15 triangles (i.e., the ring covered 15 Delaunay triangles while propagating the wave outside the given cluster).

Table 5. Experimental results for TSP Art library tests

<table>
<thead>
<tr>
<th>Test with</th>
<th>Length of the initial solution ((%))</th>
<th>Length of optimized solution ((%))</th>
<th>Run time (min)</th>
<th>Length of the best solution ((%))</th>
<th>Tour quality (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mona-L. 100K</td>
<td>5758998</td>
<td>5757516</td>
<td>121</td>
<td>5757191</td>
<td>0.006</td>
</tr>
<tr>
<td>vangogh120K</td>
<td>6545620</td>
<td>6544127</td>
<td>178</td>
<td>6543622</td>
<td>0.008</td>
</tr>
<tr>
<td>venus140K</td>
<td>6812666</td>
<td>6811271</td>
<td>213</td>
<td>6810696</td>
<td>0.008</td>
</tr>
<tr>
<td>pareja160K</td>
<td>7622498</td>
<td>7620636</td>
<td>229</td>
<td>7619976</td>
<td>0.008</td>
</tr>
<tr>
<td>courbet180K</td>
<td>7891519</td>
<td>7889462</td>
<td>280</td>
<td>7888759</td>
<td>0.009</td>
</tr>
<tr>
<td>earning 200K</td>
<td>8174726</td>
<td>8174507</td>
<td>295</td>
<td>8171712</td>
<td>0.034</td>
</tr>
</tbody>
</table>

The results shown in Fig. 13 demonstrate that the developed approach is suitable for large-scale TSP problems to get high quality solutions in a reasonable amount of time.

Fig. 13. Time vs. size for TSP Art tests.

Copyright © 2016 MECS

I.J. Information Technology and Computer Science, 2016, 5, 1-12
VI. CONCLUSIONS

1. The developed approach for parallelization has near linear computational complexity and is suitable for large-scale TSPs.

2. The developed methods provide a significant reduction in runtime with a slight decrease in the quality of the received solutions, when compared with the other best known methods.

3. Higher speedup is achieved by using the problem decomposition to split the problem into several independent sub-problems that are solved in parallel.

4. Several optimization algorithms have been developed and investigated. To improve the quality of the solution, it is useful to successively apply different optimization algorithms.

5. The biggest problem solved by the developed implementation contains 10^7 points. It needed 13 days time for calculations when a single CPU (Intel Xeon E5640 2.67GHz) was used. However, the same problem was solved in 28 hours by using cluster computing resources that contain 64 cores. In either case, the loss of the quality of the solution is very small (0.086% below when compared to the other best known solutions).

ACKNOWLEDGMENT

We thank the directors of the V.M. Glushkov Institute of Cybernetics of the National Academy of Science of Ukraine and the Institute for Condensed Matter Physics of Ukraine for providing cluster computing resources to perform our experiments. The work was performed in collaboration with the Vienna University of Technology and supported by the State Agency on Science, Innovations and Information of Ukraine.

REFERENCES


Copyright © 2016 MECS


Authors’ Profiles

Roman Bazylevych. DSc. Ph.D. Graduated the Lviv Polytechnic Institute in radio-engineering, Doctor of Sciences degree in computer sciences from the Leningrad Electro-technical Institute, Ph.D. degree in electrical engineering from the Lviv Polytechnic Institute. Currently is working as Full Professor at the Software Department of the Lviv Polytechnic National University and the University of Information Technology and Management in Rzeszow. He is a full member of the Shevchenko Scientific Society, Academician of the Ukrainian Academy of Engineering Sciences, fellow member of the IEE, senior member of the ACM. Conducted research at the Physical-Mechanical Institute of the Academy of Science of Ukraine, Lviv Polytechnic National University, University of California at San Diego, and Harvard University. Major fields of scientific research: computer sciences; combinatorial optimization; algorithms, NP-hard problems; design automation; VLSI partitioning, packaging, placement and routing; traveling salesman problem. Has more than 400 scientific works, including 4 scientific monographs, 2 textbooks, 11 patents; supervisor of 14 Ph.D. and D.Sc. dissertations; scientific leader of 72 research projects (grants and contracts).

Bohdan Kuz. Graduated from the Lviv Polytechnic National University in 2010 with master degree in Computer Science. During 2010-2013 – Ph.D. study in the Lviv Polytechnic National University in the area of mathematics and software for parallelization of large-scale TSP. In 2012, he had scholarship at the Bordeaux University (supervisor Dr. R. Dupas). Currently works for an IT company. Major fields of scientific research: software technologies, algorithms, combinatorial optimization. Has 12 scientific publications.

Roman Kutelmakh, Ph.D. Graduated from the Lviv Polytechnic National University in 2005 with master degree in Computer Science. In 2011 has finished post-graduate study and defended Ph.D. dissertation “Mathematical Methods and Software for Solving Large-Scale Traveling Salesman Problem” in the Lviv Polytechnic National University. In 2008, he had scholarship at the Bordeaux University (supervisor Dr. R. Dupas). Major fields of scientific research: algorithm analysis and design, combinatorial optimization, Currently is senior lecturer at the Software Department of the Lviv Polytechnic National University (Ukraine). Has 18 scientific publications.
Rémy Dupas is professor at Bordeaux University since 2007 (PhD thesis at Valenciennes University in 1990 and associate professor from 1993 to 2006 at Artois University). He has a scientific interest in operational research and combinatorial optimization to solve problems in the field of supply chain and logistics (i.e. production planning, vehicle routing and scheduling problems). He develops modelling and solving methods which are based on linear programming, mixed integer programming and metaheuristics (i.e. genetic algorithm, variable neighbourhood search).

Bhanu Prasad received Master of Technology and Ph.D. degrees, both in computer science, from the Andhra University and the Indian Institute of Technology Madras respectively. He is currently serving as an Associate Professor in the Department of Computer and Information Sciences at Florida A&M University. His current research interests are in the areas of artificial intelligence, software engineering, and optimization problems.

Yll Haxhimusa received a Dipl.Ing degree in Electrical and Computer Engineering from the University of Prishtina, Kosova. He obtained his Ph.D. degree in Computer Science from the Vienna University of Technology in 2006. In 2007, he became a postdoctoral associate at Purdue University (USA) in the Department of Psychological Sciences. Dr Haxhimusa’s research interests span from computer vision, pattern recognition and machine learning, to human problem solving. His current interest is the usage of machine learning and pattern recognitions methods in life sciences. Dr. Haxhimusa is an IEEE and an IAPR member.


How to cite this paper: Roman Bazylevych, Bohdan Kuz, Roman Kutelmakh, Rémy Dupas, Bhanu Prasad, Yll Haxhimusa, Lubov Bazylevych, “A Parallel Ring Method for Solving a Large-scale Traveling Salesman Problem”, International Journal of Information Technology and Computer Science(IJITCS), Vol.8, No.5, pp.1-12, 2016. DOI: 10.5815/ijitcs.2016.05.01