The effect of asymmetry on traveling salesman problems

Alejandro Rodríguez1,*; Rubén Ruiz2

1 Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática, Universidad Politécnica de Valencia, Pza. Ferrándiz Carbonell 2, 03801 Alcoy, Spain.
arodriguez@doe.upv.es

2 Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática, Universidad Politécnica de Valencia, Camino de Vera s/n, 46021 Valencia, Spain.
rruiz@eio.upv.es

November 3, 2010

Abstract

Distance or cost matrixes between groups of locations are of paramount importance when solving traveling salesman problems (TSP) as well as other routing settings. Euclidean distances constitute a gross underestimation of reality. Furthermore, reality, especially in urban transportation, is highly asymmetric. This work studies the effect that asymmetry, geographical location and territory have over TSP methods. We conduct comprehensive experiments in order to assess the effects that these factors have over some of the best known algorithms for the TSP. We demonstrate that all these factors have a significant influence in solution time and quality. Furthermore, we show that the solutions obtained with Euclidean matrixes and those obtained with real distance matrixes differ significantly. Moreover, optimal solutions to TSP problems obtained with Euclidean matrixes, when calculated with real distances, are shown to be significantly worse than the optimal solutions initially obtained with real matrixes and vice-versa.

Keywords: Asymmetry, Traveling Salesman Problem, Algorithms, Geographic Information Systems

*Corresponding author. Tel/Fax: +34 96 652 84 89
1 Introduction

The Traveling Salesman Problem, or TSP for short, is one of the most well known and thoroughly studied combinatorial optimization problems (Lawler et al., 1985). The objective is to find the minimum cost (usually minimum distance) tour visiting a set of \( n \) locations, where each location is visited exactly once. The tour must start and finish at the same location. A solution to the TSP problem is represented by a permutation of the \( n \) locations and hence, the total number of solutions is \( n! \). The TSP is a well known \( \mathcal{NP} \)-Hard problem.

In routing problems, and more precisely, in the TSP, there is a distance or cost matrix between any two locations. Each element in the matrix contains the travel distance, time, or any other cost function between any two locations \( o, d \), where \( o, d \in n, o \neq d \). Usually, travel time, speeds and costs are a function of the distances between locations or nodes. As a result, carefully estimating distances between nodes is extremely important. The need for real matrixes and distances has been highlighted several times in the TSP literature (Flood, 1956), and also for Vehicle Routing Problems (CVRP) (Clarke and Wright, 1964), or for other variants as well (Toth et al., 2001).

In this work we deal with the issue of asymmetry in the distance matrix. Its main objective is to refute the generalized and accepted assumption that effective and efficient solutions to TSP instances using Euclidean distance matrices translate to equally good solutions in realistic settings with real distance matrixes. As we will demonstrate, symmetric solutions (those obtained with symmetric and Euclidean distance matrixes) have little in common with real solutions (obtained with asymmetric and real distances). What is more important, a symmetric solution, when calculated with an asymmetric and real distance matrix deteriorates greatly, resulting in a much inferior quality than a solution directly obtained with a real asymmetric matrix. Furthermore, different state-of-the-art methods for the TSP are shown to differ in effectiveness and in efficacy when tested against asymmetric real distances, compared against original performance in Euclidean settings. Some methods even stop working when faced with asymmetric matrixes. However, it is not the intention of this paper to carry out a benchmark about state-of-the-art methods. Some other factors that also affect the level of asymmetry and the performance of TSP methods, like territory, geographical location and problem size are also studied.

More precisely, this paper addresses the following research questions: What is the effect of the asymmetry over the effectiveness and efficiency of the main TSP heuristics? Is it feasible to do Asymmetric Traveling Salesman Problem (ATSP) transformations into TSP problems? How do all the factors behave for different problem sizes? What is the most adequate heuristic in each case?

The remainder of this paper is organized as follows. Section 2 further substantiates the importance of considering asymmetry in routing problems. Section 3 elaborates on the research questions and hypotheses, together with the studied factors and variables, experimental design
and computational tests. Section 4 presents a thorough analysis of the different results from many perspectives, like CPU times, quality of the solutions and quantitative and qualitative comparisons. Finally, the conclusions of this work are presented in Section 5.

2 The real world is asymmetric

Given a TSP instance with $n$ locations or nodes, the distance matrix between any possible pair of nodes $o, d$, where $o, d \in n, o \neq d$, is denoted by $C_{[n \times n]}$ and is a square matrix where the diagonal is usually disregarded. This matrix has $n \times (n - 1)$ elements with all the distances. In the vast majority of the routing literature, the locations or nodes are determined by their coordinates in a 2D plane and the distances between each pair of nodes are calculated by the simple Euclidean distance, given by the Pythagorean formula. In this case, it is straightforward to see that the distance between the nodes $o$ and $d$ is the same as the distance between $d$ and $o$, i.e., $c_{od} = c_{do}, \forall o, d \in n, o \neq d$. In this case, the matrix $C$ is a strictly upper or lower triangular matrix with $\frac{n \times (n - 1)}{2}$ elements. A slightly more elaborated approach for obtaining the matrix $C$ is to calculate the orthodromic distance between the geolocations of two nodes. Basically, the orthodromic distance is the shortest distance between any two points on the surface of a sphere, measured along a path on the surface of the sphere itself. This is often referred to as the great-circle distance. Orthodromic distances are also symmetric in nature. Note that orthodromic distances are much more accurate than Euclidean distances when measuring long distances in Earth as Euclidean distances would traverse the Earth nucleus, not considering the Earth’s curvature.

It has been known for many decades (Daganzo, 1984) that Euclidean or orthodromic distances have little resemblance to real distances between nodes or locations that are linked through transportation networks or roads. As a matter of fact, the Euclidean or orthodromic distance is a very loose and weak lower bound of the shortest path that communicates any two nodes in a transportation network. Furthermore, when one considers the nature of traffic, one-way streets and the intricate layout of most roads, it is straightforward to see that, to some degree or another, real distance matrixes are not symmetric. This degree of asymmetry cannot be easily estimated as it varies widely according to different factors. Long distances are likely to be more symmetric due to two-way highroads. However, connecting locations in the historical centers of some big cities is likely to return asymmetric distances.

The usage of Euclidean or orthodromic distances is simply motivated by the large cost and difficulty in obtaining the real distances matrix $C$. Even nowadays, one needs to calculate $n \times (n - 1)$ shortest paths, each one constituting an enormous effort as real transportation networks, for example inside a country, typically contain billions of nodes and arcs. Geographic Information Systems (GIS) and geo-spatial databases, along with their Advanced Programming Interfaces
API facilitate, to some extent, this herculean task. In any case, this possibility is relatively recent as rich GIS systems capable of doing such calculations have only existed in the mainstream market since the mid 1990s. Before this date, and since the early 1970s, researchers have tried to calculate the real distance matrix indirectly from the Euclidean or orthodromic one. For example, some researchers tried to estimate real distances after multiplying the orthodromic matrix by a given factor (Christofides and Eilon, 1969). Other works developed some functions to estimate real distances (Love and Morris, 1972). This idea was further exploited by other authors that developed distance estimation functions depending not only on the zone where nodes are located, but also on total traveled distance (Daganzo, 1984). Many problems arise when using these functions. The proposed functions have to be adjusted mathematically and empirically (which more or less implies some validation, that in turn needs some real distance matrixes). This adjustment process is objective function dependent and also depends on the precision desired. Other authors demonstrated that this adjustment process is also dependent on the territory and other characteristics like geometry of the zone, type of transportation network, orographic accidents, natural obstacles and the like (Love and Morris, 1988; Dubois and Semet, 1995). Therefore, distance estimation cannot be carried out over the basis of a single function or without a deep and careful study, including parameter adjustment. While we do not advocate that such functions are not useful in any environment (some strategic decisions with aggregated information might benefit from such functions, where some degree of approximation is accepted), we support the idea given in Love and Morris (1988) that such functions are not acceptable in real operational settings.

As commented, a significant portion of the TSP literature considers Euclidean distances without even raising the issue of real distances. There is also a rich literature on the ATSP generalization, as for example, the papers of Gouveia and Pires (1999) and Fischetti et al. (2003), among many others. However, authors do not actually study, to the best of our knowledge, in its full complexity, the effect that different degrees of asymmetry and factors affecting asymmetry have over solution methodologies. In order to cope with all these complexities, modern GIS systems must be employed (Goodchild and Kemp, 1990), together with a deep understanding of the effect of the asymmetry and other factors over the calculation of real distance matrixes and TSP resolution.

3 Studying the effect of asymmetry

As previously stated, we are interested in either confirming or refuting the following hypotheses: 1) Asymmetry conditions the effectiveness and efficiency of the main TSP heuristics. 2) The location of the nodes in the real world generates different levels of asymmetry and therefore also conditions the TSP methods. 3) It is not always feasible to do ATSP → TSP (Jonker and Volgenant, 1983) transformations for solving real ATSP problems with TSP heuristics. 4) The size of the problem
interacts with asymmetry and also affects TSP algorithms. In order to assess these hypotheses we carry out a complete comparative study of the different solutions provided by TSP methods (Vigo, 1996), with real characteristics and dimensions as commented in Fischetti et al. (2003). A large set of TSP instances is generated to this end.

All experiments carried out strictly follow the fundamental principles of scientific research, i.e., the experiments are measurable, reproducible and with enough generality and applicability, as recommended, among many others, by Montgomery (2009). A full factorial experimental design is employed, where each generated problem instance is defined by a series of factors that are further described below.

3.1 Factors and instances generated

**Territory:** It is the geographical region where the instance is located. This region is bounded by a quadrant defined by two pairs of opposed geographical coordinates (latitude and longitude). This is a qualitative ordinal factor that has been tested at three variants, of increasing size, related with the Iberian peninsula, as shown in the leftmost picture at Figure 1. The three regions are referred to as short, medium and large distance, respectively.

In the short distance, locations are placed in the geographical area of influence of a city. As a result, the minimum distances between pairs of nodes are conditioned by urban transportation networks (one-way streets, traffic circles, city center, etc.). Medium distance includes short distance plus larger distances entailing regional transportation through paths, regional roads, city communication rings, etc. Lastly, large distance territories are further conditioned by large distance roads, highways and inter-city communications.

![Figure 1: Different territories in the Iberian peninsula (left). Example of an instance with locations following a radial distribution in a large distance territory (right).](image)

**Location:** It is the placement of the nodes inside the territory. This can be random or might follow a given pattern. Three variants are defined for this nominal qualitative factor: random, grid and radial. Figure 2 shows some examples over a given territory. In the grid location
distribution, the territory is divided into square zones. The node is placed at the center of each zone, albeit slightly displaced by a random vector. Radial distribution has a central location that services the remaining \( n - 1 \) nodes, which are radially distributed at an angular equidistance equal to \( \alpha = \frac{2\pi}{n - 1} \). Figure 1 (right) shows a map with 500 radially distributed locations in a large territory.

![Example locations in random, grid, and radial distributions](image)

Figure 2: Examples of locations in random (left), grid (middle) and radial (right) distribution.

**Number of nodes:** This number \( n \) determines the size \( n \times (n - 1) \) of the matrix or, \( 2n \times 2(n - 1) \) if it is transformed (the transformation process is detailed next). It is a quantitative factor with the following 10 levels \( n = \{50, 100, 150, \ldots, 500\} \).

**Symmetry:** For each generated instance, the distance matrix \( C \) is calculated in different ways. This qualitative nominal factor considers strictly symmetric or asymmetric matrixes with the following studied variants:

- **Orthodromic:** It is the symmetric matrix with great-circle distances.
- **Asymmetric:** Asymmetric matrix where the distances have been calculated with the aid of a GIS, i.e., distances are actually the shortest distance between locations as per the real network of roads and streets.
- **Minimum arc from each pair:** It is a symmetric matrix where distances have been extracted from the asymmetric matrix in a special way. Given any two distinct nodes \( o \) and \( d \), the distance for the matrix is the minimum of the two ways, i.e., the distance satisfies \( \min(c_{od}, c_{do}) \). This results in a symmetric matrix.
- **Maximum arc from each pair.** Similar to the previous one but taking the maximum of the two ways: \( \max(c_{od}, c_{do}) \).
- **Transformed:** A symmetric matrix is constructed from the asymmetric one using a well-known mathematical transformation due to [Jonker and Volgenant (1983)](Jonker1983). This transformation is correlated with the number of nodes in the instance as the transformation multiplies the size of the distance matrix by a factor of four. Each location or node is split
into two nodes, one real, and a second virtual node. The distance between a real node and
its corresponding virtual sibling is set to a very small favorable cost (usually \(-\infty\)). This
results in real and virtual nodes to be consecutively placed in the final TSP tour. The
original asymmetric “from”-“to” ways are assigned to distances between real nodes in the
transformed matrix whereas original asymmetric “to”-“from” distances (i.e., the way back
distances) are assigned to the virtual nodes. All other possible distances are assigned a
very unfavorable value \((+\infty)\).

All four factors, together with their corresponding levels or variants are gathered in Table 1:

<table>
<thead>
<tr>
<th>Territory (T)</th>
<th>Location (L)</th>
<th>Number of nodes (n)</th>
<th>Symmetry (M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short distance</td>
<td>Random</td>
<td>50</td>
<td>Orthodromic (O)</td>
</tr>
<tr>
<td>Medium distance</td>
<td>Grid</td>
<td>100</td>
<td>Asymmetric (A)</td>
</tr>
<tr>
<td>Large distance</td>
<td>Radial</td>
<td>150</td>
<td>Minimum arc (P)</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>500</td>
<td>Maximum arc (G)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Transformed (T)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| 3 | 3 | 10 | 5 |

Table 1: Factors for the instances along with their levels and variants.

As we can see, the last row of Table 1 contains the total number of levels or variants for
each factor. Since we employ a full factorial experimental design, we have \(3 \times 3 \times 10 \times 5 = 450\)
treatments after combining all levels or variants. For each treatment, five different instances are
generated, for a grand total of 2,250 TSP instances. All these instances are publicly available
at [http://soa.iti.es](http://soa.iti.es).

3.2 Response variables

As we will detail later, several state-of-the-art TSP methods are used for solving the proposed
instances. Using Design of Experiments allows to study the effect that each considered factor
(including the different algorithms) have over one or more response variables.

Solutions obtained after solving each instance are analyzed mainly at two levels: quantitative
(mainly tour length) and qualitative (sequence of nodes or locations in the tour). As regards
the last qualitative assessment, the literature is marred with papers that propose indicators
for measuring the differences between solution objects, as for example Schiavinotto and Stützle
(2007). In our case, measuring the differences between two TSP tours is commonly carried out by
counting the number of \(k - opt\) movements that are needed to transform one tour \(s\) into another
\(s'\). This needs a non polynomial CPU time as a function of \(n\). Therefore, we employ simpler
measures of a distance \(d\) between two tours or \(d(s, s')\):

**Relative percentage deviation from the best solution found** \(\Delta S^*_i\): It is the relative
deviation (in percentage) of the tour length obtained after solving a given TSP instance \(i\) with
algorithm $A$ ($S_{i,A}$) from the lowest known tour length for that instance ($S_i^*$). It is calculated as follows:

$$\Delta S_i^* = \frac{S_{i,A} - S_i^*}{S_i^*} \cdot 100$$

**Hamming distance $dH$:** It is a well known indicator that measures the differences between vectors, proposed by [Hamming (1950)](#). Basically, it takes two tours $s$ and $s'$ and adds 1 to the indicator counter each time a position in the tour is occupied by different nodes at both tours. For example, given $s = [2, 5, 3, 1, 4, 6]$ and $s' = [1, 2, 3, 4, 5, 6]$, the Hamming distance is 4. There is a problem as regards the TSP since the relative order of nodes in the tour is as important as their absolute positions. Take a second example $s = [6, 1, 2, 3, 4, 5]$ and $s' = [1, 2, 3, 4, 5, 6]$. In this case, the Hamming distance is 6, even though the route is almost the same (the only difference being the starting/ending node. However, this indicator is simple to calculate (it just requires $O(n)$ steps) it is easy to understand and to interpret.

**Adjacency distance $dA$:** Together with the Hamming distance, it makes sense to measure also the number of equal adjacent nodes between two routes $s$ and $s'$, where the nodes need not be located in the same absolute positions at the two tours. More specifically, this is achieved by checking if the arc between nodes $e$ and $e+1$ at solution $s$ –s($e, e+1$)– exists in any place of sequence $s'$. As a result, the adjacency distance counts the number of distinct arcs between two tours, with the maximum possible distance being $n+1$. For example, given $s = [1, 2, 3, 4, 5, 6]$ and $s' = [1, 6, 2, 3, 4, 5]$, $dA = 3$, since arcs (1, 6) (6, 2) (5, 1) of $s'$ are not present at $s$. Using efficient data structures, $dA$ can be calculated in $O(n)$ steps.

**CPU time:** It is the real elapsed CPU time that was needed when solving a given instance with an algorithm. This excludes input/output operations as well as all other system overheads, as detailed in [Alba (2006)](#).

**Asymmetry in distance matrices:** We are particularly interested in measuring the asymmetry degree on matrices. Simply speaking, a matrix is asymmetric if it exists at least one pair $o, d$ such that $c_{od} \neq c_{do}$, where $o, d \in n, o \neq d$. Furthermore, this even true if $c_{od} = c_{do} + \epsilon$, for any arbitrarily low value of $\epsilon$. Obviously, this binary asymmetry indicator is not very informative and more precise indicators are needed. We employ the following: **Alfa ($\alpha$):** It indicates the asymmetry degree by counting the number of asymmetric pairs of distances (pairs $o, d$ that satisfy $c_{od} \neq c_{do}, o, d \in n, o \neq d$) over the total number of pairs, using the $x_{a}(o,d)$ definition and expression (2) below. $\alpha$ takes values in the $[0\%, 100\%]$ interval. **Delta ($\delta$):** It measures the asymmetry degree in more detail by actually looking at how different are asymmetric pairs (in distance). It is calculated with expression (3) below. **Weight:** It just sums all the distances of the $C$ matrix, i.e., $\sum_{o=1}^{n} \sum_{d=1, o \neq d}^{n} c_{od}$. **Average weight (Weight):** It relates the weight with the number of arcs.

8
\[ x_a(o, d) = \begin{cases} 
0 & \text{if } c_{od} = c_{do} \\
1 & \text{if } c_{od} \neq c_{do} 
\end{cases} \]

\[ \alpha = 2 \sum_{n=1}^{n} (x_a) \cdot 100 \]  

(2)

\[ \delta_a = \left| c_{od} - c_{do} \right| + \min(c_{od}, c_{do}) \cdot \frac{100 - 100}{\min(c_{od}, c_{do})} \quad \forall o, d \in n, \ o \neq d \]  

(3)

### 3.3 Solution process

For solving all instances, we employ a high performance computing cluster with 30 blades, each one containing 16 GBytes of RAM memory and two Intel XEON E5420 processors running at 2.5 GHz. Note that each processor has 4 physical computing cores (8 per blade).

At this stage, it is worth mentioning the sheer computation effort needed for calculating real distance matrixes (all instances where M=A), especially when compared against orthodromic matrixes. 450 instances in the set of 2,250 contain real distances. These have been calculated by doing a humongous number of shortest route requests between pairs of nodes to Google Maps. This took 196.5 single blade equivalent CPU days. This is in stark contrast with the 21 seconds needed for calculating the same matrixes but with orthodromic distances.

A direct outcome of this computational effort is a large set of 450 ATSP instances where distances are actually real, corresponding to current transportation networks in Spain, following all previous factors already mentioned in earlier sections. This set is complementary to the well known TSPLIB95 dataset where only 19 synthetic ATSP instances can be found. These ATSP instances have random integer distances at each arc with \( n \) sizes between 17 and 443. As indicated, these instances are publicly available.

Each instance is solved with a series of TSP heuristics:

- Nearest neighbor algorithm (A=NN) as described in [Flood](1956). A simple heuristic, yet with a reasonable performance.

- 2-Opt heuristic ([Rego and Glover](2002)) (A=2O). A well known simple local search method.

- Concorde TSP solver\(^1\) (A=CO). A very powerful state-of-the-art exact branch-and-cut algorithm for the TSP. It is described in [Applegate et al.](2002).


- Improved Lin-Kernighan of [Helsgaun](2000) (A=HE). This is currently considered as one of the state-of-the-art methods for solving the TSP.

[^1]: http://www.tsp.gatech.edu/concorde.html
As we can see, the selection of TSP heuristics is motivated either by simplicity or by current state-of-the-art performance. Note that not all studied heuristics are capable of working over asymmetric matrixes. For example, the LK and CO methods are specifically designed for the TSP and not for the ATSP (Applegate et al., 2006). In these cases, the transformed matrix \( M = T \) is employed instead of the real asymmetric one. This results in 2,250 instances \( \times 5 \) algorithms \( -450 \) asymmetric matrixes \( \times 2 \) non-ATSP heuristics (LK and CO) = 10,350 computational experiments. All these experiments needed 556 single blade equivalent CPU hours. No CPU time limit was imposed to any algorithm.

4 Analysis of results

All results are supported by statistical analyses. We mainly use the multifactor Analysis of Variance (ANOVA) technique where we control all studied factors. Three different groups of response variables are considered: CPU times needed by the algorithms, quantitative and qualitative comparison of symmetric (TSP) and asymmetric (ATSP) tours. All results are detailed in the following sections. Since the ANOVA is a parametric technique, one needs to check the three main hypotheses which are normality, homocedasticity and independence of the residuals. The residuals resulting from the experiment were analyzed and no serious deviations were observed.

4.1 CPU times

Some of the most interesting results are observed when analyzing the CPU times needed by the algorithms capable of solving ATSP problems. The resulting ANOVA table is given in Table 2.

At a 95% confidence level \( (\alpha = 0.05) \), all single factors and 8 double factor interactions are statistically significant. Among significant factors, importance is observed by the magnitude of the \( F \)-Ratio. For example, the \( F \)-Ratio of the factor “Algorithm” is no less than 23,548.98. This means that the differences among the different algorithms generate 23,548.98 more variance than the variance obtained within each algorithm. Therefore, the type of algorithm has a very strong and statistically significant influence over the CPU time.

The ANOVA technique mainly points out statistical significance. For further understanding of the behavior of any studied factor, we need descriptive plots. We have included plots with points and smoothed lines for comparing the CPU time as a function of the size of the matrixes for all combinations of Symmetry (types of matrixes) and Algorithms factors. All these plots are shown in Figure 3 where the X-axis gives the size of the matrix \( (n) \) and the Y-axis the CPU time in seconds, with a logarithmic scale. Each row in the plot corresponds to a type of matrix and each column to an algorithm. Note that there are no plots for algorithms LK and CO for asymmetric matrixes.

We confirm that the type of algorithm has a strong influence over CPU time. As expected, The improved Lin-Kernighan method of Helsgaun (HE) and Concorde (CO) are the most com-
### Table 2: Analysis of Variance (ANOVA) for CPU time response variable and ATSP algorithms (M≠T, A≠CO and A≠LK).

<table>
<thead>
<tr>
<th>Source</th>
<th>Sum of squares</th>
<th>Degrees of freedom</th>
<th>Mean square</th>
<th>F-Ratio</th>
<th>p-Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main Effects</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A: Territory</td>
<td>233.72</td>
<td>2</td>
<td>116.86</td>
<td>5.00</td>
<td>0.0067</td>
</tr>
<tr>
<td>B: Location</td>
<td>1176.5</td>
<td>2</td>
<td>588.25</td>
<td>25.19</td>
<td>0.0000</td>
</tr>
<tr>
<td>C: Symmetry</td>
<td>7075.48</td>
<td>3</td>
<td>2358.49</td>
<td>101.01</td>
<td>0.0000</td>
</tr>
<tr>
<td>D: n</td>
<td>734028</td>
<td>9</td>
<td>81558.7</td>
<td>3492.86</td>
<td>0.0000</td>
</tr>
<tr>
<td>E: Algorithm</td>
<td>1.099E6</td>
<td>2</td>
<td>549871</td>
<td>23548.98</td>
<td>0.0000</td>
</tr>
<tr>
<td><strong>Interactions</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AB</td>
<td>38.31</td>
<td>4</td>
<td>9.58</td>
<td>0.41</td>
<td>0.8014</td>
</tr>
<tr>
<td>AC</td>
<td>468.87</td>
<td>6</td>
<td>78.14</td>
<td>3.35</td>
<td>0.0027</td>
</tr>
<tr>
<td>AD</td>
<td>569.39</td>
<td>18</td>
<td>31.63</td>
<td>1.35</td>
<td>0.1434</td>
</tr>
<tr>
<td>AE</td>
<td>2193.9</td>
<td>4</td>
<td>548.48</td>
<td>23.49</td>
<td>0.0000</td>
</tr>
<tr>
<td>BC</td>
<td>446.59</td>
<td>6</td>
<td>74.43</td>
<td>3.19</td>
<td>0.0040</td>
</tr>
<tr>
<td>BD</td>
<td>2097.47</td>
<td>18</td>
<td>116.53</td>
<td>4.99</td>
<td>0.0000</td>
</tr>
<tr>
<td>BE</td>
<td>2166.35</td>
<td>4</td>
<td>541.59</td>
<td>23.19</td>
<td>0.0000</td>
</tr>
<tr>
<td>CD</td>
<td>8234.88</td>
<td>27</td>
<td>304.99</td>
<td>13.06</td>
<td>0.0000</td>
</tr>
<tr>
<td>CE</td>
<td>87919.9</td>
<td>6</td>
<td>14653.3</td>
<td>627.55</td>
<td>0.0000</td>
</tr>
<tr>
<td>DE</td>
<td>917929</td>
<td>18</td>
<td>50996.0</td>
<td>2183.98</td>
<td>0.0000</td>
</tr>
<tr>
<td>Residual</td>
<td>123055</td>
<td>5270</td>
<td>23.35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total (corrected)</td>
<td>2.987E6</td>
<td></td>
<td>5399</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Figure 3: CPU time behavior for all algorithms, types and sizes of matrixes.
putationally demanding algorithms. The size of the matrix also affects CPU time directly and exponentially and this is the case for all types of matrixes, symmetric and asymmetric and for all algorithms. Also expected is the matrix transformation process of matrixes (M=T), which results in enormous CPU time increases. This is a logical result that validates the whole experiment, as the size of the original asymmetric matrix is multiplied by four in the transformation process. However, and as shown in Figure 3 with symmetric matrixes, HE is actually slower than CO (about three times slower). This is an unexpected result as CO is an exact procedure and HE, albeit extremely effective, cannot guarantee optimality. The matrix transformation (M=T) affects much more CO than HE as the CPU time increases approximately by a factor of 7. The problem is that CO only works with symmetric matrixes and the transformation is the only possible way of dealing with asymmetric problems. The collorally is that CO is far more sensible to the size of the TSP to solve.

As regards the other studied factors, Grid locations result in higher CPU times for the exact CO method, specially when compared to the Random and Radial locations. This is depicted in the means plot of Figure 4. This Figure contains the interaction between two factors for all matrixes except the asymmetric ones. The two studied factors are the CO and HE algorithms and the Location. The means are plotted in the middle of 95% Honest Significant Difference (HSD) Tukey confidence intervals. Overlapping intervals denote that the means contained within them are not statistically different. As we can see, CPU times increase sharply for the CO method for Grid locations and this difference is statistically significant. From our perspective, this result is a primer in the routing literature. To the best of our knowledge, there are no reported studies that analyze how the distribution of the nodes or clients affect the CPU times of state-of-the-art methods. While HE is unaffected by this factor, we see that CO CPU times more than double by just changing how the nodes are distributed.

![Figure 4: Means plot with Honest Significant Difference (HSD) Tukey confidence intervals (95%) for the interaction between Location and Algorithm factors, where the response variable is CPU time (M≠A).](image)

Other interesting findings affect the Territory factor. Again, the most affected algorithm is
CO. Figure 5 shows the interaction between HE and CO algorithms and the Territory factor. While HE remains unaffected, CO needs significantly less CPU time for solving short range problems. A possible explanation is that in short distances there is more variability in the distances between nodes and possibly this helps in the branch-and-cut process of CO.

![Figure 5](image)

Figure 5: Means plot with Honest Significant Difference (HSD) Tukey confidence intervals (95%) for the interaction between Territory and Algorithm factors, where the response variable is CPU time ($M_A$).

Finally, CPU time is affected, on average, by the Symmetry factor (type of distance matrix). If we remove the transformed matrix ($M=T$) which we have already seen increases CPU times by orders of magnitude, the result obtained is shown at Figure 6. We see that there are no statistically significant differences between the $M=P$ and $M=G$ matrixes. Recall that these represent symmetric matrixes where the distances are the minimum and maximum distances, respectively, between the from-to and to-from asymmetric distances in the matrix. This means that the differences in CPU time cannot be attributed to the magnitude of the distances, but rather to the differences in the distances themselves. We also observe how asymmetric matrixes ($M=A$) need significantly more CPU time than regular orthodromic matrixes ($M=O$).

![Figure 6](image)

Figure 6: Means plot with Honest Significant Difference (HSD) Tukey confidence intervals (95%) for the Symmetry factor (type of matrix), where the response variable is CPU time ($M \neq T$, $A \neq CO$ and $A \neq LK$).
4.2 Quality of solutions

It has to be reminded that the objective at this step is not to measure which algorithm, among the tested ones, is the best. The focus is rather on studying how the considered factors affect the quality of the solutions provided by the algorithms. We start by analyzing the response variable $\Delta S^*$, defined by expression (1) in previous sections. Table 3 provides the number of times that each algorithm provides the best solution ($N_*$), $S^*$, and the corresponding rate (% $S^*$) under three different settings. The second and third columns indicate matrixes $M=(O, P, G)$ (1,350 experiments per algorithm). The fourth, fifth, sixth and seventh columns indicate asymmetric ($M=A$) and transformed ($M=T$) cases, respectively, with 450 experiments per algorithm.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$M=(O,P,G)$</th>
<th>$M=A$</th>
<th>$M=T$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$N_*$</td>
<td>$% S^*$</td>
<td>$N_*$</td>
</tr>
<tr>
<td>NN</td>
<td>0</td>
<td>0.0%</td>
<td>0</td>
</tr>
<tr>
<td>2O</td>
<td>3</td>
<td>0.2%</td>
<td>2</td>
</tr>
<tr>
<td>LK</td>
<td>628</td>
<td>46.6%</td>
<td>-</td>
</tr>
<tr>
<td>CO</td>
<td>1350</td>
<td>100%</td>
<td>-</td>
</tr>
<tr>
<td>HE</td>
<td>930</td>
<td>68.9%</td>
<td>448</td>
</tr>
</tbody>
</table>

Table 3: Number of best solutions and success rate for the studied algorithms and types of matrices.

As expected, CO always produces the optimum solution for the 1,350 symmetric instances. Expectedly, HE’s success rate is high at almost a 70% and much better than LK’s at almost 47%. The simple heuristics NN and 2O are rarely successful. For asymmetric matrixes ($M=A$), HE is clearly dominant and no further results can be drawn from our comparison since LK and CO do not accept asymmetric matrixes. Most surprisingly, CO does not obtain the optimum solutions in all cases for $M=T$. An obvious explanation is that in the transformation process, some values in the matrix are $-\infty$ or $+\infty$ and this creates numerical instability problems inside CO that result in small deviations from the true optimum solution. With this result, we can now conclude that not only CO needs an exponentially greater CPU time for transformed matrixes, but also that the results cannot be trusted. Naturally, with modifications inside the CO code, there is the possibility that transformed matrixes could be considered without glitches. Another interesting outcome is that for transformed matrixes, LK outperforms HE. However, the transformation process is actually not needed for HE and we cannot conclude that LK is preferred over HE for transformed matrixes.

Numerous statistical analyses were performed in order to check the influence of the studied factors over the quality of the solutions. Multiple Analyses of Variance experiments were performed, which are not fully detailed due to space restrictions. These are summarized at Table 4. It has to be noted that since no maximum CPU time was given to all tested algorithms, the...
Table 4: Different Analysis of Variance (ANOVA) experiments performed over the $\Delta S_i^*$ response variable.

effect of the different studied factors over solution quality is about 1% or less (contrary to the previous observed effects on CPU time). While this might be seen as a marginal effect, it has to be reminded that in the TSP state-of-the-art literature, publications and new results are often disputed with improvements of less than 2% in solution quality (Helsgaun, 2000). However, almost all factors resulted statistically significant in all tests carried out.

Table 5 shows average $\Delta S_i^*$ values for the different tested algorithms as a function of the type of matrix (symmetry factor). Again, we see the large deterioration in HE and CO with transformed matrixes ($M=T$). Whereas for all other matrixes, either CO or HE clearly dominate. Once again we see that for transformed matrixes, it is even better to use LK than HE.

Table 5: Average $\Delta S_i^*$ values according to Algorithm and Symmetry (type of matrix) factors.

Similar results, but in this case for the Territory factor, are shown at Table 6. It is interesting to see that short distances have a strong influence on the quality of solutions, even for the best performers CO and HE. Furthermore, an although not shown, this effect is observed for all matrix sizes and specially for transformed matrixes.

We now analyze the behavior of the different algorithms against the Location factor in Table 7. As shown, no overly strong effects are observed (albeit all differences are statistically significant).

Lastly, it is worth mentioning that matrix size has a very small impact on $\Delta S_i^*$ values. The effect is less than 0.06% in the worst case. Figure 7 shows the averages of the Symmetry and Algorithms factors (excluding $M=T$). The horizontal axis shows the size of the matrix $n$ and the vertical axis shows the percentage deviation $\Delta S_i^*$ over the best solution. The following Figure 8
shows a similar plot containing just the M=T matrixes.

### 4.3 Quantitative and qualitative assessment

After studying the different matrixes’ asymmetry degree and all other studied factors, we found that the asymmetry degree of a matrix ($\delta$) and the average weight ($\text{Weight}$) of the different distances in the matrix are strongly related (Rodríguez and Ruiz, 2010). If there is a relation between symmetric and asymmetric matrixes in the form of an increased average weight, it is logical to think that the symmetric solution of the TSP could be “augmented” in order to carefully estimate the real ATSP solution (as regards the total tour length). This is needed since, as we have already stated, the TSP tour length is a loose lower bound of the real ATSP tour length. Similarly, it is important to check the tour length of the TSP solution, when calculated with the ATSP matrix and viceversa. In order to check all these questions we use the following indicators, which are strongly based on the previously defined response variables.

\[ \Delta \text{ATSP} : \text{It is the percentage increase of the tour length of the ATSP solution as regards the TSP solution:} \]

\[ \Delta \text{ATSP} = \frac{\text{ATSP} - \text{TSP}}{\text{TSP}} \cdot 100 \]  \hspace{1cm} (4)

\text{TSP} is the tour length of the symmetric problem, calculated with symmetric orthodromic matrixes. \text{ATSP} is the asymmetric problem tour length, calculated with real distance matrixes (M=A). Note that algorithms LK and CO cannot solve the ATSP. In these cases the transformed matrix (M=T) is used instead.
Figure 7: Average $\Delta S^*_i$ values according to the size of the matrix $n$, algorithms CO, HE and LK and Symmetry factors.

Figure 8: Average $\Delta S^*_i$ values according to the size of the matrix $n$, algorithms CO, HE and LK and transformed matrixes (M=T).
TSP_A: The TSP solution is calculated with asymmetric matrixes. i.e., we take the solution of a TSP problem and recalculate it with the real distances. Obviously, the tour length will increase (TSP_A ≥ TSP).

ATSP_O: It is the opposite case as TSP_A. The ATSP solution is calculated with the symmetric matrix.

ΔTSPA: It is the percentage increase of TSP_A against ATSP. It could be positive or negative.

ΔATSP_O: It is the percentage increase or decrease of ATSP_O against TSP.

ΔdH: It is the percentage of differences in the TSP solution against ATSP. Values close to 100% qualitatively indicate that the TSP solution is very different from the ATSP. It is based on the previously defined Hamming distance dH:

\[ \Delta dH = \frac{dH}{n} \cdot 100 \] (5)

ΔdA: It is the percentage of different arcs, over the total arcs, that the TSP sequence has over the ATSP solution. It is based on the previous adjacency distance dA:

\[ \Delta dA = \frac{dA}{n + 1} \cdot 100 \] (6)

We calculate all previous indicators for all experiments, namely 3 territories ×3 locations ×10 different matrix sizes ×5 different algorithms ×5 replicates which results in 2,250 data. The results are sound. The average ΔATSP indicator reaches a value of 81.6%. This indicates a huge difference between the ATSP and TSP solutions. Note that the minimum observed value for this indicator is an already large 32.9% (the maximum being a whooping 196.9%). The frequency distributions of the ΔATSP values are given as an histogram in Figure 9 (left). It is observed that in a large percentage of the cases, the increase is between 50% and 100%. Figure 9 (right) shows a second histogram, this time for ΔTSPA. The distribution is clearly skewed towards positive values, with an average of 12.4%. Exactly, 37.7% of the cases show differences equal or larger than 10%.

A very strong result, specially with the second histogram, is that there is a large difference between solving symmetric and asymmetric problems. The commonly accepted idea that orthodromic or Euclidean distances for solving the TSP are valid and that after all, one can later calculate real distances with the solution is utterly wrong. As shown, on average, there are differences that amount to a 12.35%. Note that these differences are extremely large for TSP problems where algorithms fight in the state-of-the-art segment for less than 2% improvements.

As regards ΔATSP_O, the distribution (not shown) is again positive, with an average of 15.3%. In a 54.62% of the cases, the differences are equal or greater than 10%. Once again, it is shown that it is not the same solving an asymmetric ATSP and measuring the tour length of the sequence...
In our opinion, this is the main contribution of this research work. From our perspective, the commonly accepted assumption of considering Euclidean distances does not hold when solutions are calculated with real asymmetric distances. The TSP solutions deteriorate enormously when calculated with real matrixes. Furthermore, there is little guarantee that a good algorithms for the TSP will work equally good for the ATSP. Often, the degradation in performance will be significantly greater than the observed differences between competing methods.

Another significant result is to qualitatively observe the big differences between the sequences obtained with symmetric TSP and asymmetric ATSP problems. The average $\Delta dA$, with a value of 71.7% indicates that the symmetric sequences are almost entirely different from the asymmetric ones. Values are even greater if one uses the Hamming distance $\Delta dH$. The minimum value for the $\Delta dA$ value is as high as 9%. This means that, in the best case, a full 9% of the sequence is different. Figure 10 (left) shows the frequency distribution of $\Delta dH$. Note how it is heavily skewed towards values very close to 100%. Figure 10 (right) shows the corresponding histogram for $\Delta dA$.

4.4 Some examples

In this section we graphically depict a couple of examples. The purpose is to graphically show the large differences between symmetric and asymmetric solutions. Only two examples have been randomly selected due to obvious space constraints.
CC-0503HE. The first example corresponds to instance CC-0503. A total of 50 nodes are grid located in a short distance territory. We compare the best solution obtained with algorithm HE. There are huge differences between the symmetric optimal tour length of 244.2 km. versus the 423.3 km. of the different optimal tour length calculated with the asymmetric distances. Therefore, $\Delta ATSP = 73.32\%$. Additionally, we have that $\Delta dA = 90.20\%$. Figure 11 has the two optimum solutions superimposed. The symmetric in blue and the real one in red.

MR-0504CO. In this second example we show the results of the instance MR-0504 solved with CO algorithm. We have again 50 nodes radially distributed in a medium distance territory. The total tour length in the symmetric tour is 1,369.7 km. versus 2,097.9 km. for the asymmetric tour length. This results in a $\Delta ATSP$ of 53.17\%, with a $\Delta dA$ of 98.04\%. Figure 12 shows the
5 Conclusions and further work

In this work we have studied the effect that the asymmetry has over the solution process of the TSP. We have shown that solving the TSP differs much from solving the ATSP at so many levels: tour length, adjacency of nodes, hamming distance and CPU time. Furthermore, all these differences are strongly affected by other studied factors. The main objective has been to show that solving a TSP and later calculating the real tour length with real distances is not a viable solution process. During the research, thousands of instances have been solved with some of the best well known (including some state-of-the-art) algorithms for the TSP.

We have been able to confirm, as expected, that the algorithm used strongly affects the CPU time. HE and CO are the most computationally costly methods, much more than the other simplistic heuristics. We have also shown that the Territory, Location and size of the problem factors all affect the different methods in a sound and statistically significant way. For symmetric matrixes, locations in grid have a significant effect on CPU times, resulting in harder to solve problems. This effect is amplified with transformed matrixes and affects much more CO than HE. It has been demonstrated that the Territory has an impact on CPU time, specially for short distance territories. For symmetric transformed matrixes, the Territory factor is relevant as regards the quality of the solution. This effect is observed for all matrix sizes. A relevant conclusion of this study is that the transformation process has a profound effect over CPU times. Furthermore, this process has shown to be not entirely feasible for algorithms HE and, specially, for CO.

If one closely compares HE and CO for symmetric matrixes, we observe that HE is about three
times slower than CO. This is unexpected, since CO is an exact branch-and-cut method and HE a (powerful) heuristic. However, the transformation process increases the computation time more than sevenfold, and affects CO much more. The result is that asymmetry has a deep impact over algorithms, either from the quality of the solutions standpoint, or from the CPU time whenever matrix transformation is needed.

Comprehensive statistical experiments further demonstrate that there is an inverse relation between the Territory factor and the average $\Delta ATSP$ indicator value. This confirms that there are quantitative differences between the $TSP$ and $ATSP$ solutions. The differences between the two solutions are smaller for large distances than for short distances. Furthermore, we found relations between the Location factor and the $\Delta ATSP$ indicator. In this case, the existing differences are greater as the size of the problems grows and they are greater for radially and randomly placed locations than for grid ones. The size of the matrix also conditions the differences between the $TSP$ and $ATSP$ solutions.

It has not been the objective of this work to compare state-of-the-art methods neither for the $TSP$ nor for the $ATSP$. The main objective has been to evaluate of different factors and degrees of symmetry affect these problems. Further work stems from the possibility of providing effective methods for calculating real asymmetric travel matrixes, as this imposes today a clear entry barrier for those researchers not willing to use the typical Euclidean matrixes. Extending this study to more complex problems, like for the Capacitated Vehicle Routing Problem CVRP or for the Heterogeneous Fleet variant ($HFCVRP$) is of interest to see if more realistic routing problems are equally affected by asymmetry.

Acknowledgements

The authors would like to warmly thank Prof. Keld Helsgaun for facilitating the code of his algorithm. This work is partially funded by the Spanish Ministry of Science and Innovation, under the project “SMPA - Advanced Parallel Multiobjective Sequencing: Practical and Theoretical Advances” with reference DPI2008-03511/DPI. The authors should also thank the IMPIVA - Institute for the Small and Medium Valencian Enterprise, for the project OSC with references IMIDIC/2008/137, IMIDIC/2009/198 and IMIDIC/2010/175 and the Polytechnic University of Valencia, for the project PPAR with reference 3147.

References


