International Journal of Geo-Information

MDPI

Article

Spatial Transformation of Equality – Generalized Travelling Salesman Problem to Travelling Salesman Problem

Mohammed Zia [1,2,*,†], Ziyadin Cakir [2,3,†] and Dursun Zafer Seker [1,†]

[1] Geomatics Engineering Department, Istanbul Technical University, Maslak, Istanbul 34469, Turkey; seker@itu.edu.tr
[2] National Innovation and Research Center for Geographical Information Technologies, Maslak, Istanbul 34469, Turkey; zia@itu.edu.tr
[3] Department of Geology, Faculty of Mines, Istanbul Technical University, Maslak, Istanbul 34469, Turkey
* Correspondence: mohammed.zia33@gmail.com; Tel.: +90-537-343-8708
† Current address: Istanbul Technical University, Maslak, Istanbul 34469, Turkey.

Received: 9 February 2018; Accepted: 23 February 2018; Published: 15 March 2018

Abstract: The Equality-Generalized Travelling Salesman Problem (E-GTSP), which is an extension of the Travelling Salesman Problem (TSP), is stated as follows: given groups of points within a city, like banks, supermarkets, etc., find a minimum cost Hamiltonian cycle that visits each group exactly once. It can model many real-life combinatorial optimization scenarios more efficiently than TSP. This study presents five spatially driven search-algorithms for possible transformation of E-GTSP to TSP by considering the spatial spread of points in a given urban city. Presented algorithms are tested over 15 different cities, classified by their street-network's fractal-dimension. Obtained results denote that the R-Search algorithm, which selects the points from each group based on their radial separation with respect to the start–end point, is the best search criterion for any E-GTSP to TSP conversion modelled for a city street network. An 8.8% length error has been reported for this algorithm.

Keywords: generalized travelling salesman problem; shortest route; combinatorial optimization; OpenStreetMap

1. Introduction

The Travelling Salesman Problem (TSP) is one of the most well-known and extensively studied combinatorial optimization problems by far. It has been used as a benchmark problem for new urban and navigation developments for decades. It asks to find the shortest (in terms of length, time, or custom cost) route that visits each vertex in a given set exactly once before returning to the starting vertex. Formally, it could be stated as follows: given a directed/undirected graph $G = (V, E)$ with set of vertices V and set of weighted edges E, find the shortest path between start vertex s and end vertex e (e could be same as s for closed path) that visits each vertex of a given set $V' \subseteq V - \{s, e\}$ exactly once. It is equivalent to finding the minimum-cost Hamiltonian cycle in G [1]. It has many applications in ranging areas of geo-spatial sciences and GIS, like in vehicle routing, urban communication networking, public transport sequencing and scheduling, to name a few (please read [2]). It has always been a great source of attraction from various other disciplines too, especially during the last three decades [3].

The Generalized-TSP (GTSP) or Set-TSP or Travelling Politician Problem is one useful exemplary of the Travelling Purchaser Problem. It is one practical extension of the TSP, first introduced by [4], where the set of vertices V is further segmented into n number of groups and it asks to find a minimum-cost route visting at least one vertex from each group before reaching the destination. For almost all inherently hierarchical real-world urban and navigation problems, it offers a more precise model

than TSP. GTSP could mathematically be defined as follows: Let $G = (V, E)$ be a graph where $V = \{v_1, v_2, ..., v_n\}$ is the set of vertices, $E = \{(v_i, v_j) \mid i \neq j; v_i, v_j \in V\}$ is the edge set, and $W = \{w_{ij}\}$ is the non-negative cost or weightage defined on E. If E is undirected, then directions become irrelevant, i.e., $(v_i, v_j) = (v_j, v_i)$. Furthermore, V is partitioned into x mutually exclusive and exhaustive groups such that $V^g = \{V_1, V_2, ..., V_x\}$ and $V = V_1 \cup V_2 \cup ... \cup V_x$ with $V_\alpha \cap V_\beta =$ for all $\alpha, \beta = 1, 2, ..., x$ and $\alpha \neq \beta$. It asks to determine the shortest Hamiltonian route that passes through each group at least once (introduced independently by [4–6]) or exactly once (introduced by [7,8]). If the matrix W is symmetrical, i.e., $w_{ij} = w_{ji}$ for all $i, j = 1, 2, ..., n$ and $i \neq j$, the problem is prefixed as symmetric; otherwise, it is asymmetric. This results in many vertices from each group left to be visited. The exactly once variant of the GTSP is also known as the Equality-GTSP or E-GTSP [9], where the shortest route contains exactly one vertex, i.e., station, from each group in V^g. The E-GTSP is an NP-hard problem [10] as it reduces down to the famous TSP (also NP-hard) whenever individual groups become singleton, i.e., $(|V_\alpha| = 1 \; \forall \; \alpha = 1, 2, ..., x)$. Figure 1a represents one possible closed Hamiltonian cycle, also known as g-tour, that visits exactly one vertex from each group before returning to the starting vertex. In Figure 1b, the cycle is open as the start and end groups are discrete.

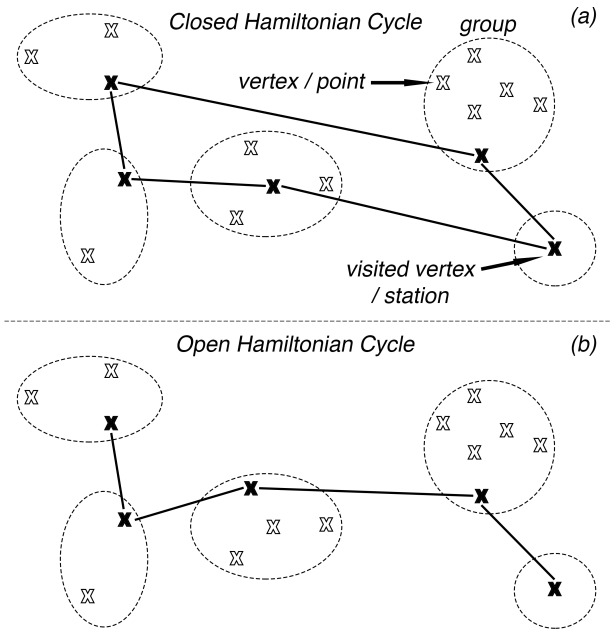

Figure 1. Representation of one possible (**a**) closed and (**b**) open Hamiltonian cycle in a given symmetric E-GTSP instance. Here, the group-counts are equal to 5, and the total number of vertices are equal to 16; in other words, $x = 5$ and $n = 16$, respectively.

A discussion on how the GTSP can be used as one versatile and elegant tool to model different classes of combinatorial optimization problems like the covering tour problem, the material flow system design, the post-box collection problem, the stochastic vehicle routing problem, and the arc routing problem has been provided by [11]. All these different use-cases show the importance of GTSP and E-GTSP in advanced GIS and city infrastructure planning and development. They have prime relevance in location-based problems, urban planning, postal routing problem, logistics problems, telecommunication problems, and railway track optimization problems. [8,12] have also discussed similar applications in detail.

Problem Encountered

In this study, the authors are primarily interested in the E-GTSP to TSP transformation, taking the location based services, vehicle routing and urban planning aspects into account. It should be noted that there already exists a large variety of exact and heuristic methods to solve a TSP [13–15] like the state-of-the-art Lin–Kernighan–Helsgaun TSP solver [9]. Since the underlying model represents an urban street setup, the authors have tried to think of this transformation from a spatial perspective. Five different possible search algorithms, motivated by the nearest neighbour search model [16] and Voronoi model [17], are suggested and tested on a real-world OpenStreetMap (OSM) street-network dataset to find the optimal search criterion for fast E-GTSP to TSP transformation. The idea is an extension of the authors' previous work [18]. This conversion is especially important for cases where the network or graph or cost-matrix of an urban street-network is generated on the fly (cases where cost is a function of time, like function of traffic congestion on the road). Different group-counts (i.e., $|V^g| = 1, 2, 3, 4, 5$) are employed to test all proposed algorithms for different complexity levels. Generated results are presented in graphical form and discussed in depth in the Results and Discussion section, Section 5.

To the best of authors' knowledge, this kind of spatial transformation of a given E-GTSP to TSP is the first of its kind and no related work is available in peer-reviewed literature online. It is believed that this study will further open research paradigms to improve existing search algorithms or to derive a better one for complex urban development and navigation problems. In the following sections, the proposed search algorithms are explained and tested on OSM derived road network data-set. Finally, a commentary on the results is provided in the Conclusions section.

2. Transformation of the E-GTSP to TSP

The GTSP was first introduced by [4–6] through a record balancing problem that was aroused in the computer design. It is one of the few optimization problems that has extensively been studied in the past [8,19]. Researchers [8,20–24] have attempted to transform this into some efficient TSPs by exploiting the dynamic programming techniques [25,26], disintegrating a complex problem into a few simpler sub-problems). The shortcoming of this transformation, however, is that it dramatically increases the dimension of the problem, i.e., the dimension of the matrix representing the GTSP, thereby affecting the time and space consumption. Therefore, although theoretically it is possible to solve a given GTSP by converting it into corresponding TSP, the new increased problem size makes it computationally expensive to solve.

An E-GTSP solution of a vehicle-routing model is dependent upon the following two **decisions** in written order:

1 Selection of a subset of vertices (V^s), also termed as stations, such that $V^s \subseteq V$ and $V^s \cap V_\alpha = 1$ for all $\alpha = 1, 2, ..., x$. *Note* that $|V^s| = |V^g|$.

2 Calculation of the minimum-cost Hamiltonian cycle in subgraph $G^s = (V^s, E^s)$.

Since in our E-GTSP model the start and end vertices are different, the calculated g-tour is going to be an open one (Figure 1b). Presented search algorithms do not increase the size of the problem by increasing the number of vertices or edges. Considering a vertex's spatial distribution with respect to the start and end vertex, it is easier to filter-out distant and possibly sub-optimal vertices from each group. In this article, *points* represent all possible vertices of V including the start and end vertex, *group* represents each set of vertex from V exhibiting one particular attribute, and *station* represents all selected points from each group (Figure 1). *Points* could be assumed as all supermarket, bank, etc. locations within a city, *group* could be assumed as a set of certain attribute like *group* of supermarket, *group* of bank, etc., and *station* could be assumed as those supermarket, bank, etc. locations that users should visit for least costly routes.

3. Methodology

The main objective of this article is to answer **decision 1** (Section 2) for an E-GTSP model that asks to find the shortest route from start to end points that visits each group exactly once. In urban planning and navigation scenarios, these groups could represent all locations of the bank, pharmacy, cafeteria, etc. Five different search criteria to select the exactly one point from each group depending upon the start and end point are discussed in this section (Figure 2). Although a number of such algorithms are possible with little tweaks, the authors believe that the selected ones are mutually absolute and cover a whole breadth of different search possibilities. Since this kind of spatial transformation is the first of its kind, no other spatial approach for E-GTSP to TSP transformation has been discussed elsewhere. Primarily, all five algorithms consider here the spatial spread of points in a given urban set-up in order to select the optimal one from each group for the shortest route estimation.

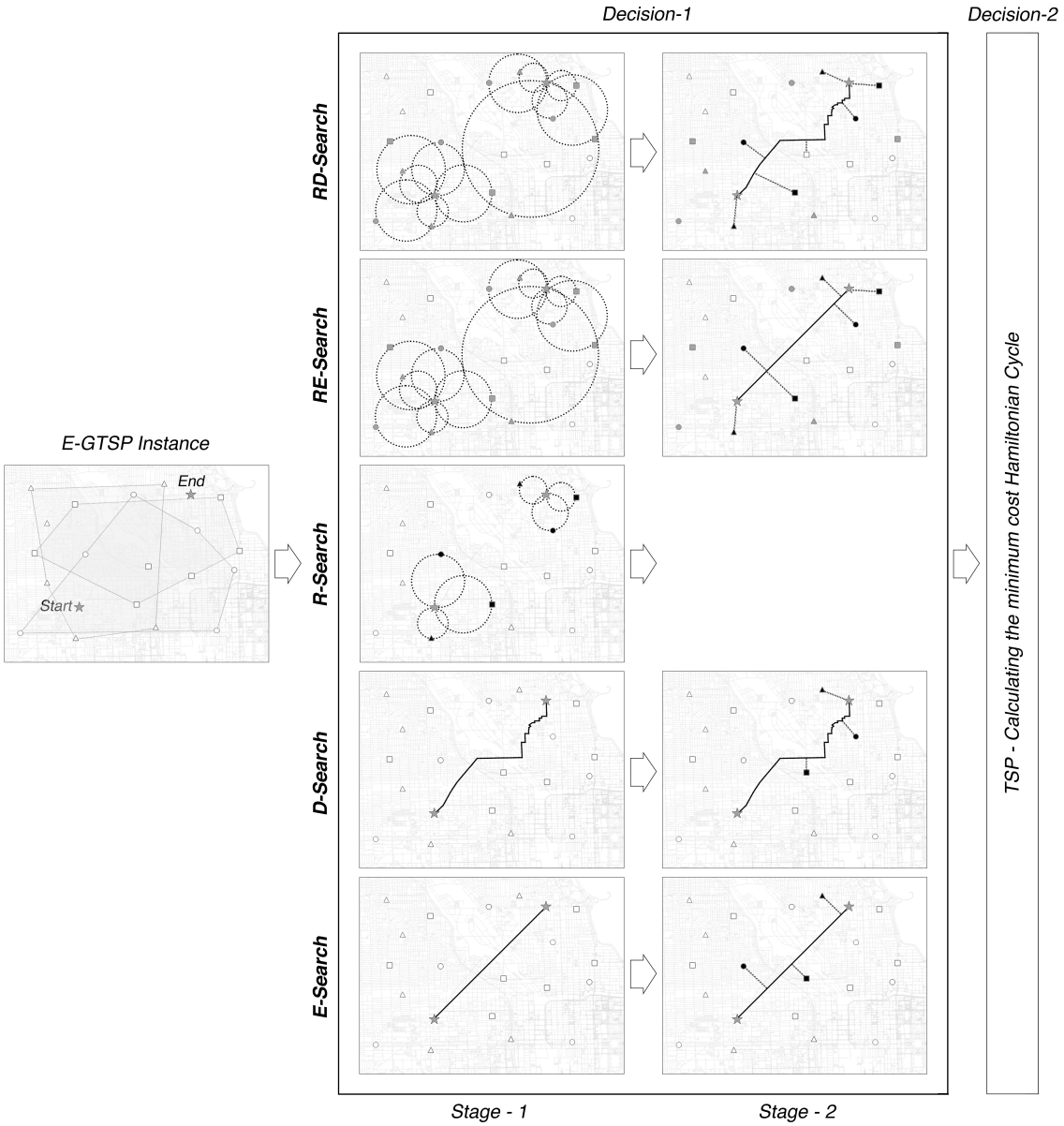

Figure 2. Diagrammatic representation of five different proposed search algorithms. Three groups of different points, marked by oval, triangular, and rectangular markers, are used to show an E-GTSP instance. It should be noted that, for R-Search, RE-Search, and RD-Search, there are two stations from each group after **decision 1** (Section 2), thus generating eight ($2 \times 2 \times 2$) different TSPs for **decision 2**.

3.1. E-Search and D-Search

Euclidean-Search (E-Search) is the first search criterion that is based upon the Euclidean separation of start and end points in an instance. As shown in Figure 2, the basic approach here is to connect both the start and end points by a straight line and select the closest point from each group with respect to this line. It involves two stages before reducing the E-GTSP to TSP. It is $O(n)$, where $n = |V|$.

Dijkstra-Search (D-Search) on the other hand, involves the calculation of Dijkstra route, instead of the Euclidean line, between given start and end points before selecting the closest one from each group with respect to this route (Figure 2). Computationally, it is more expensive than the E-Search, with $O(n^2)$. Algorithm1 is a pseudo-code of the two search algorithms above.

Algorithm 1 E-Search and **D-Search** pseudo-code

Require: Terminal points S and F, $G = (V, E)$ representing urban street-network, & $V^g = \{V_1, V_2, ..., V_x\}$, i.e., points' groups set, where $V_i \subseteq V$ & $V_i \cap V_j = \forall\, i, j = 1, 2, ..., x$ & $i \neq j$.

procedure E-GTSP $\Rightarrow$ TSP
  if Algorithm 1 $\widehat{=}$ **E-Search then** Draw SF **Euclidean** Line
  end if
  if Algorithm 1 $\widehat{=}$ **D-Search then** Calculate SF **Dijkstra** Route
  end if
  Initialize an empty container $C1$
  for $V_i \in V^g \;\; \forall \;\; i = 1, 2, .., x$ **do**
    Initialize an empty container $C2$.
    for $v_m \in V_i \;\; \forall \;\; m = 1, 2, .., |V_i|$ **do**
      $C2 \leftarrow v_m - SF$ shortest Euclidean distance
    end for
    return $C2$
    $C1 \leftarrow v \;\widehat{=}\; v_m - SF \in C2$, with the smallest Euclidean distance
  end for
  return $C1$
end procedure
procedure TSP
  Calculate the minimum-cost Hamiltonian cycle induced by $C1$. *Note*: $C1 = V^s$.
end procedure

▷ $\widehat{=}$ means corresponds to

▷ $\leftarrow$ means append

3.2. R-Search

Radial-Search (R-Search) is the third way to select the stations based upon their radial distance from both the start and end points. The closest point from each group is selected twice, making $|V^s| = 2 \times x$ (Section 2), generating a total 2^x different TSPs. This makes **decision 2** computationally expensive (Figure 2). There is only one stage in **decision 1**. It involves an O-complexity of $O(n)$ and the Algorithm 2 represents its pseudo-code. It should be noted that, unlike the E-Search and D-Search, it also evaluates the points lying outside the proximity of the start–end region (white region in Figure 6).

Algorithm 2 R-Search pseudo-code

Require: Terminal points S and F, $G = (V, E)$ representing urban street-network, & $V^g = \{V_1, V_2, ..., V_x\}$, i.e., points groups set, where $V_i \subseteq V$ & $V_i \cap V_j = \forall\ i, j = 1, 2, ..., x$ & $i \neq j$.

procedure E-GTSP $\Rightarrow$ TSP
  Initialize an empty container $C1$
  for $V_i \in V^g\ \ \forall\ \ i = 1, 2, .., x$ **do**
    Initialize an empty container $C2$.
    for $v_m \in V_i\ \ \forall\ \ m = 1, 2, .., |V_i|$ **do**
      $C2 \leftarrow v_m - S$ Euclidean distance
      $C2 \leftarrow v_m - F$ Euclidean distance
    end for
    return $C2$
    $C1 \leftarrow v \mathrel{\widehat{=}} v_m - S$ & $v_m - F \in C2$, with the smallest Euclidean distance
  end for
  return $C1$
end procedure
procedure TSP
  Calculate the minimum-cost Hamiltonian cycle for all pairs (i.e., 2^x) induced by $C1$.
  Select the pair with shortest minimum-cost cycle.
end procedure

3.3. RE-Search and RD-Search

Finally, the last two search algorithms are the hybrid of R-Search and E-Search (RE-Search), and R-Search and D-Search (RD-Search). They first ask to find the radially closest two points from each group with respect to both the start and end points independently, making $|V^s| = 2 \times 2 \times x$. Then, they reduce the size of the V^s to half by selecting the closest stations from both ends with respect to the Euclidean line (for RE-Search) or Dijskstra route (for RD-Search), respectively (Figure 2). They have the complexities of $O(n)$ and $O(n^2)$, respectively. The second stage, thus, leads to a total of 2^x number of TSPs, similar to the R-Search. Algorithm 3 represents their pseudo-codes.

Once all desirable stations are selected, i.e., optimal V^s, from each group for minimum-cost open Hamiltonian route, it is solved by the Simulated Annealing [27] TSP Solver. Although in this study the authors have used this algorithm to solve the reduced TSP instance (**decision 2**), one is free to use any solver, as it will equally affect all five of the search algorithms. The presented results in the Results and Discussion section, Section 5, are more relative in nature.

Algorithm 3 RE-Search and **RD-Search** pseudo-code

Require: Terminal points S and F, $G = (V, E)$ representing urban street-network, & $V^g = \{V_1, V_2, ..., V_x\}$, i.e., points groups set, where $V_i \subseteq V$ & $V_i \cap V_j = \forall\ i, j = 1, 2, ..., x$ & $i \neq j$.

procedure E-GTSP $\Rightarrow$ TSP
  Initialize an empty container $C1$
  for $V_i \in V^g \ \ \forall \ \ i = 1, 2, .., x$ **do**
    Initialize an empty container $C2$.
    for $v_m \in V_i \ \ \forall \ \ m = 1, 2, .., |V_i|$ **do**
      $C2 \leftarrow v_m - S$ Euclidean distance
      $C2 \leftarrow v_m - F$ Euclidean distance
    end for
    return $C2$
    $C1 \leftarrow$ Two $v \mathrel{\widehat{=}} v_m - S$ & $v_m - F \in C2$, with least Euclidean distances
  end for
  return $C1$
  if Algorithm 3 $\mathrel{\widehat{=}}$ **RE-Search then** Run **E-Search** (Algorithm 1), treating $C1 \equiv V^g$
  end if
  if Algorithm 3 $\mathrel{\widehat{=}}$ **RD-Search then** Run **D-Search** (Algorithm 1), treating $C1 \equiv V^g$
  end if
  Estimate $C1^f$ (Filtered $C1$ after **E-Search** or **D-Search**)
end procedure
procedure TSP
  Calculate the minimum-cost Hamiltonian cycle for all pairs (i.e., 2^x) induced by $C1^f$.
  Select the pair with shortest minimum-cost cycle.
end procedure

4. Study Area and Data-Set Used

In order to test all five proposed search algorithms, a real-world OSM street-network dataset is used. Fifteen cities belonging to the five different fractal dimension (frac-D) bins are selected for analysis out of the total 210 number of cities worldwide [28] (Figure 3). An increasing frac-D of an urban set-up represents its increasing road density, with 1D representing regions with only one road segment and 2D representing regions completely filled-up with road segments. Bin size in Figure 3 has intentionally been kept small, i.e., 0.1, for finer fractally resolved classes. The Fractal Dimension Calculator is used to calculate the dimension of a city [29]. Table 1 contains all analyzed cities, sorted with increasing value of frac-D, along with corresponding decisive attributes derived from the OSM vector dataset.

OSM dataset is downloaded from its Overpass API [30] covering the bounding box of each selected city [31]. All 105 start–end pairs are selected on the basis of the most visited locations by people, i.e., business center, tourist spot, and residential area. Five different groups of points representing bank, supermarket, cafe, pharmacy, and petrol-pump are taken from the OSM dataset to picture a near real-world E-GTSP routing instance. In total, only five such groups are selected from the OSM dataset for this study in order not to increase the brute-force processing time to find the optimal route. The whole processing is done in Python language. Instead of using the sample instances from existing GTSP libraries, like the GTSPLIB [32], the authors have created their own E-GTSP instances from downloaded OSM vector data. This provides an opportunity to test the presented algorithms on specific urban planning and navigation scenarios. The authors have also provided the data/meta-data of all tested instances along with an optimal V^s set to encourage other researchers to carry out similar studies on OSM [33]. Five E-GTSP instances with different $|V^g|$ values are used for each selected city in order to run the discussed algorithms for various complexities. Key observations of the best algorithm and its behavior are discussed in Section 5.

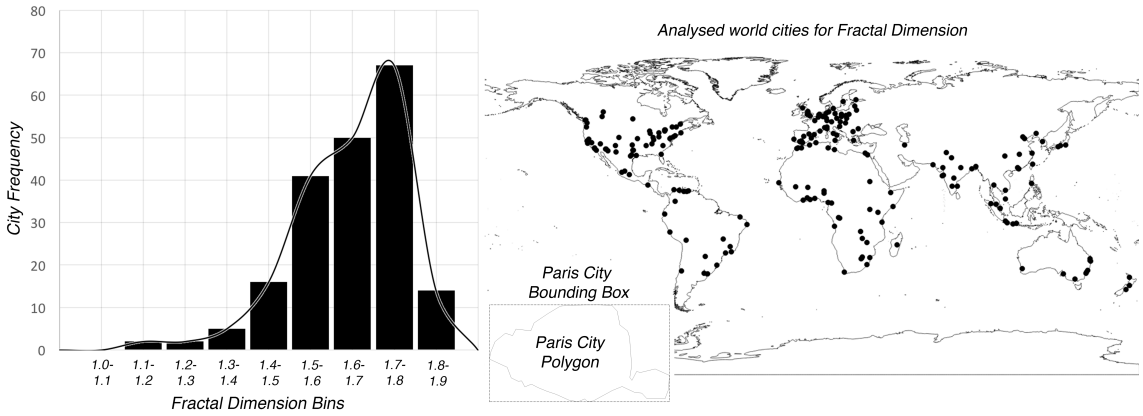

Figure 3. Histogram showing the frequency of cities (marked on the world map **right**) for each Fractal-Dimension bin. Increase in the dimension represents an increase in the road-density. Three cities are selected from each of the top five bins to test all proposed search algorithms. Each city-polygon's bounding box is used to download the corresponding XML data from OSM Overpass API.

5. Results and Discussion

In order to test the five proposed search algorithms above for efficient E-GTSP to TSP transformation, 15 cities are selected based on their different level of street-network patterns and density (quantified by frac-D, Table 1), for which the data was obtained from the OSM API. For each city, five different instances, i.e., $|V^s|$, modeling various levels of E-GTSP complexity, are created and tested. Figure 4a is a 3D-plot between the different search algorithms used, the different numbers of stations to be visited, i.e., $|V^g|$, and the different types of street-networks. Here, each colored circle represents the average fractional error (average of the fractional errors coming out from all 105 analyzed start–end pairs) in the E-GTSP route-length estimated with respect to the optimal route (by brute-force). There are, in total, 375 (15 cities $\times$ 5 group-counts $\times$ 5 algorithms) colored circles, with black circles representing errors of more than 20%. It is clear that, irrespective of the choice of a given search algorithm, the percentage error in route length increases with higher $|V^s|$ instances (marked by a big white arrow Figure 4a). It is an expected behavior in a combinatorial optimization problem. Figure 4b represents five graphs between different search algorithms and different type of street-networks, belonging to different $|V^s|$ values. Black-boxes represent the lowest fractional error for that row. It can be seen that, for lower $|V^s|$ values, the D-Search algorithm gives the lowest percentage error for the most number of cities, irrespective of their frac-Ds. However, this performance drifts away towards the R-Search algorithm for higher group-counts. It is an interesting observation that suggests that, for complex E-GTSP scenarios, it is better to select the optimal points that are radially close with respect to the start and end points. Another important observation is that, in spite of being hybrid, the performance of RE-Search and RD-Search was quite low. It should be noted that high computational complexity does not always mean better precision. Quantitatively, D-Search is the best one for all analyzed instances. However, it is important to understand how different algorithms have performed qualitatively (Figure 5).

Figure 5a contains five different plots displaying the average fractional error for each city on the y-axis, ordered according to Table 1, for different $|V^s|$. Each colored circle represents one distinct search algorithm. In order to make the plots comparable, the red (D-Search) and green (R-Search) circles are connected. It can be seen that, for $|V^s| = 1$, the red line is almost always below the green one (Figure 5a), showing its out-performance. However, for a higher $|V^s|$ count, this observation is reversed. We have also already mentioned this observation in Figure 4b. Figure 5a allows us to compare them absolutely. Figure 5b is a summation of the average fractional errors for all $|V^s|$ instances for each city. It is clear that the green line lies below the red one for most of the cities. This makes R-Search, accuracy-wise, the best possible overall search-algorithm for any given E-GTSP to TSP transformation,

particularly for urban planning and navigation use-cases. In vehicle routing problems, absolute route length acts as one vital attribute for any route selection process and, therefore, the R-Search approach should be considered. Figure 9 gives the percentage route-length error for different search-algorithms. R-Search gives the lowest average error of 8.8% when tested on a real-world dataset. It would be interesting to compare the performance of it with other approaches that are developed by [25,26] on a similar data-set.

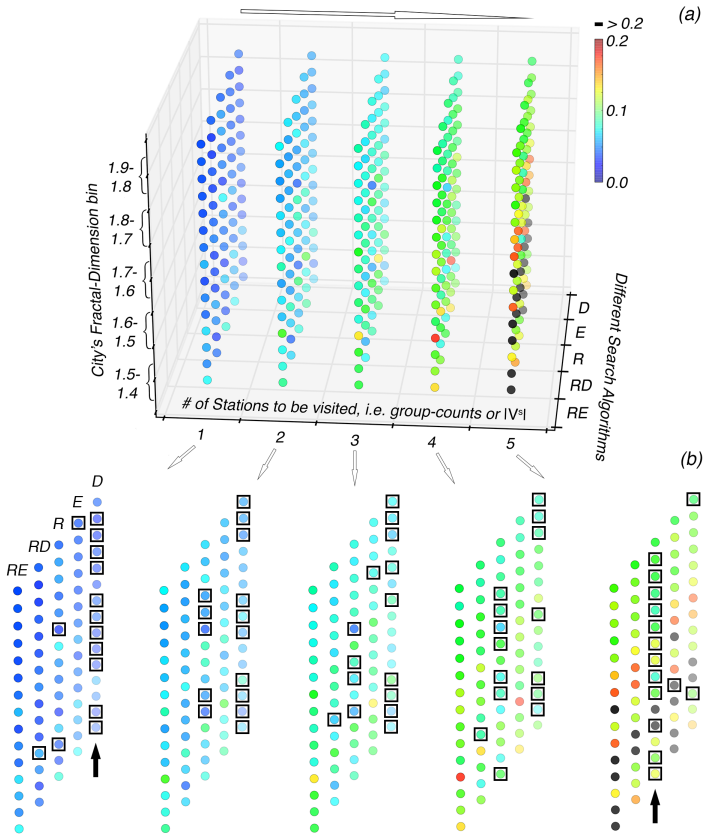

Figure 4. 3D-graph between Different Search Algorithms, Number of Stations to be visited, i.e., $|V^s|$, and City's Fractal-Dimension bin. Each bin represents three cities taken from Figure 3. Color represents the average fractional error in route-length coming out from each algorithm with respect to the optimal one (estimated by brute-force), with the color black having errors of more than 20%. (**a**) It should be noted that, with an increase in the $|V^s|$ for a given E-GTSP instance, the error also gets increased irrespective of the choice of the algorithm (marked by big arrow). (**b**) The graph is sliced-down for each group-count, individually. For each city, the algorithm with the least errors is marked by black box. It is clear that, for small $|V^s|$ values, the D-Search is better, and, as one increases the number of stations, the R-Search outperforms the other, marked by solid-arrows. Overall, the R-Search is advised for any urban based E-GTSP to TSP conversion (Section 5).

Figure 6 gives one possible explanation of the performance of R-Search for higher $|V^s|$ instances. The grey area is the proximity region of a given start–end points pair, while the white area is the region outside it. D-Search is quite efficient in selecting points lying close to the underlying Dijskstra route. Since this route terminates at the start–end points, points that lie behind them (white region, Figure 6) do not get considered for stage 2 (**decision 1**) (Figure 2). On the other hand, R-Search remains unbiased towards points for their regions (white/grey) (Figure 2). Although this leads to the total 2^x number of TSP routes, the R-Search approach is better for white-regioned points. The y-axis in Figure 6 represents the average fraction of the total number of optimal stations for each $|V^s|$ instance for white and grey regions. It should be noted that the number of optimal stations that fall within the grey-area gets lower

with increase in complexity ($|V^s|$). This shows that, with increase in the number of group-counts, the more and more optimal stations come out from the white-region. This makes the performance of R-Search better. This is an interesting observation, which shows that different features/building functions in an urban city generally spread-out homogeneously. A continuous drop of the curve in Figure 6 right helps us generalize the performance of R-Search for complex instances too.

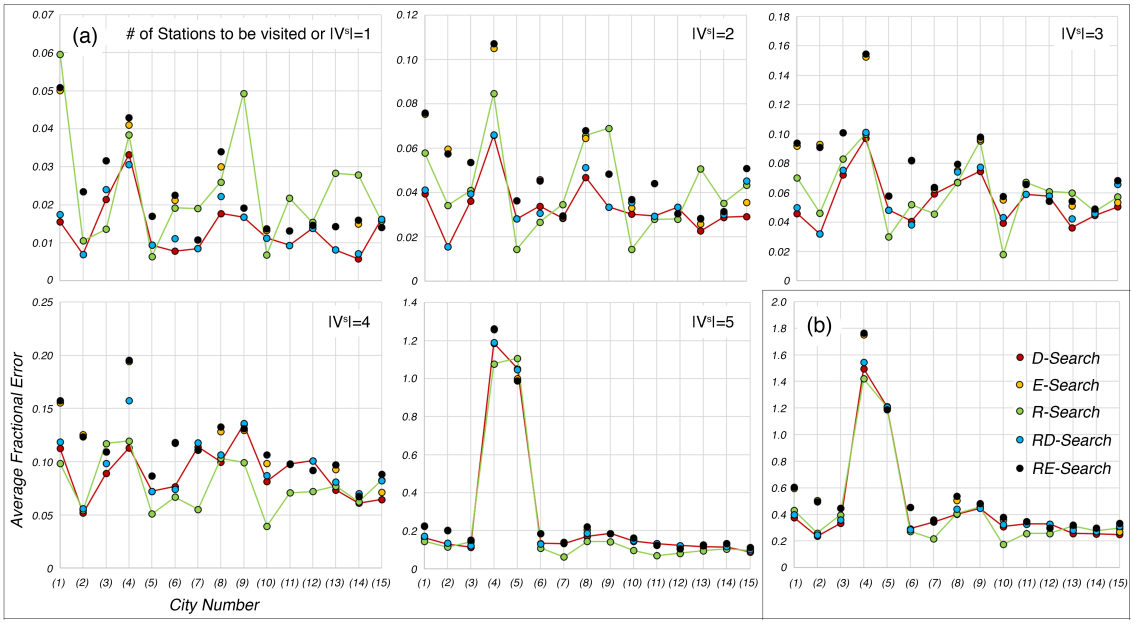

Figure 5. (**a**) scatter plot between the Average Fractional Error and the City Number (representing city, Table 1) for all $|V^s|$ instances. Nodes belonging to the D-Search and R-Search are joined together to allow comparison. It can be seen that, for lower $|V^s|$ instances, the D-Search is better than the R-Search, and vice-versa for higher $|V^s|$ instances, as also reported in Figure 4; (**b**) the y-axis represents the summation of Average Fractional Error from all $|V^s|$ instances. It is said that the R-Search is overall the best search criterion to perform a geospatial E-GTSP to TSP conversion.

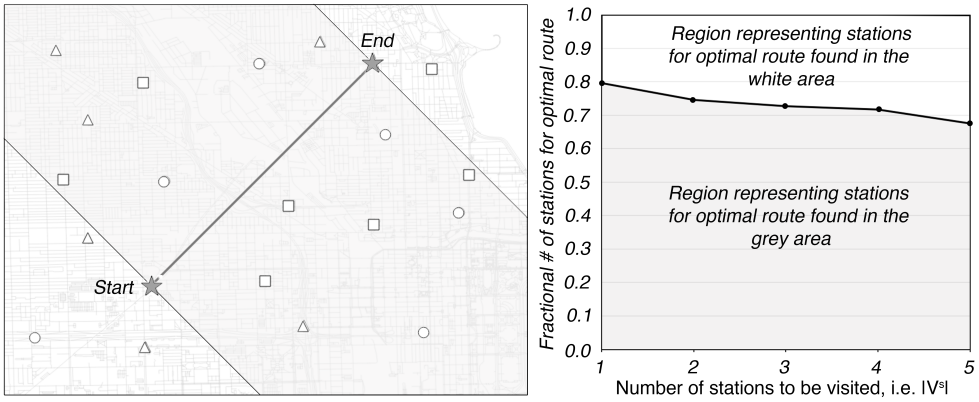

Figure 6. Possible explanation of the out-performance of R-Search for increased complexity. It can be seen that the total number of optimal stations within the white region increases with increase in the complexity of the model (**left plot**). Since R-Search also selects optimal stations from the white region, it performs better for higher $|V^s|$ instances (**right plot**).

In order to observe the D-Search and R-Search performance for increasing start–end points' separation, a plot is generated between their estimated route length and optimal length (Figure 7).

The graph represents all 105 tested routes from all 15 cities for different complexities. The key observation here is the fanning-out behavior of all data points with increase in optimal route length. As one increases the best route length, the estimated value by both the algorithms gets farther away from the mean-line (solid) (Figure 7). Although only the $|V^s| = 5$ scenario is presented here, a similar observation has been done for other instances too. This behavior is expected, as almost all routing algorithms get error-prone for distant start–end point pairs. Furthermore, in a given city's road-network, the optimal route length of a complex E-GTSP instance might reach the order of a few couple of tens of kilometers. This shows the necessity to find the best possible solution in a timely fashion of a given TSP problem, as it will directly affect the time and money of the user.

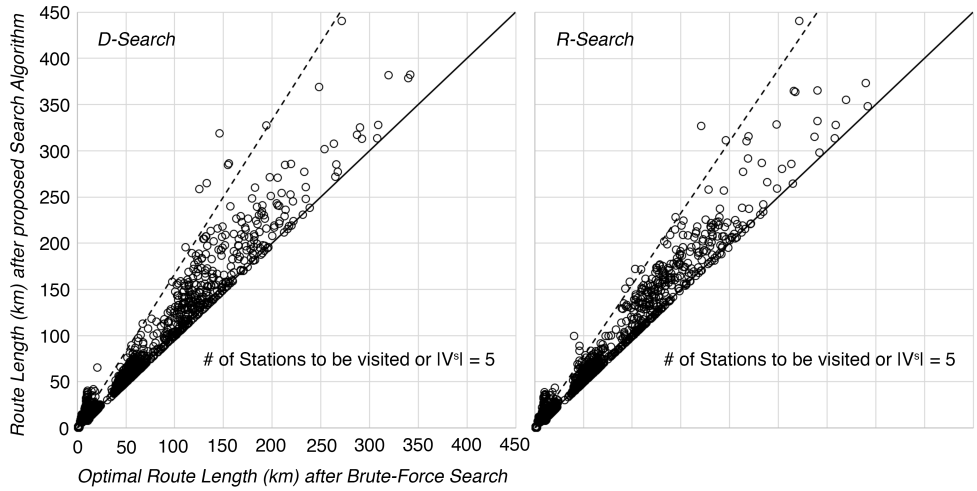

Figure 7. Graph comparing the absolute route lengths for brute-force and D-Search/R-Search approaches. It can be seen that the intensity of the error increases with increase in the length of the trip. Although only $|V^s| = 5$ instance is plotted here, a similar fanning-out behavior (dashed- and solid-line) is observed for other cases too. Severe errors are expected for long trips irrespective of the choice of algorithm or complexity of model.

Finally, a plot between the start–end points' separation and their optimal route length value for different instances are created (Figure 8). Instead of plotting the individual data-points on a graph, the authors have sketched their regions with differently styled lines for better visualization and explanation. With increase in the complexity of an instance ($|V^s|$) for a given city's E-GTSP model, Brussels (Belgium) in this case, corresponding polygons drift away (marked by a solid arrow) from the mean-line towards some higher y-value. On the right side of Figure 8, there are two plots showing the slope of the trendline and R^2 value with respect to the $|V^s|$. The trendline here belongs to each data-set that is sketched by the polygon. Each curve exemplifies one city model in Figure 8. Decreasing slope of the trendline with increasing complexity shows the saturation of optimal route length value. It means that, for larger group-counts for the given E-GTSP model, the optimal route length becomes more independent from its start–end point geo-location. This observation is quite novel and shows that once an optimal route length of a given start–end pair for a complex instance is calculated, it could be extrapolated for other pairs too. Similar to the trendline, its R^2 value also decreases with complexity. Furthermore, the decreased R^2 value shows the complex street-networks of real-world cities. For complex instances, therefore, an amalgamation with heuristic approaches is advised, which is also necessary because a new route estimation for each scenario every time is computationally expensive.

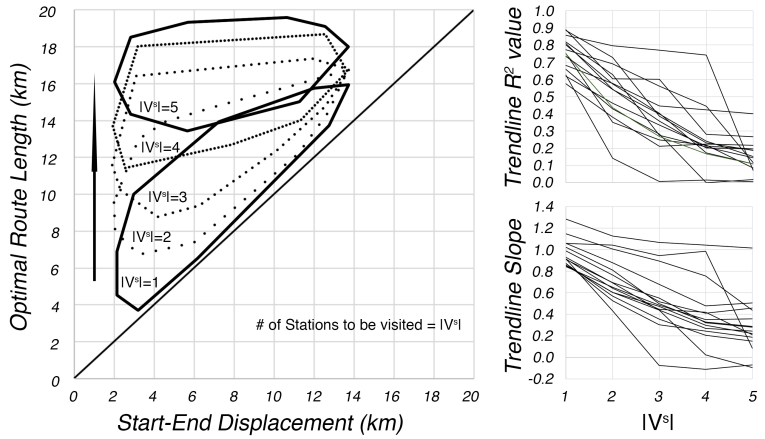

Figure 8. Graph (**left**) between the start–end points' geo-separation and corresponding optimal route length for all five group-counts (for Brussels). Note that a polygon is sketched to show the spread of each data point. It should be noted how polygons drift away from the mean-line for increased complexity. On the **right**, we have slope and R^2 values of trendlines induced by each $|V^s|$ instance for all 15 cities, where there are decreases with increases in complexity. The decreasing R^2 and slope value along the x-axis depicts the complex networking of urban street-roads, and shows how optimal route length gets similar irrespective of the start–end points' spatial separation.

Table 1. General statistics of all selected cities (Figure 3).

ID	City (Country)	Fractal Dimension	Area (km^2)	OSM # Vertex	OSM ρ Vertex	OSM # Edge	OSM ρ Edge	$\sum$Edge (km)
1	Hargeisa (Somalia)	1.400	42,332	7267	0.17	10,839	0.26	7782
2	Antananarivo (Madagascar)	1.413	42,291	26,955	0.64	33,188	0.78	10,455
3	La Paz (Bolivia)	1.432	14,454	55,239	3.82	83,209	5.76	11,012
4	Nairobi (Kenya)	1.511	84	3,386	40.3	4469	53.2	482
5	Mexico City (Mexico)	1.517	1153	6839	5.93	7,819	6.78	1241
6	Las Vegas (USA)	1.537	81,440	154,289	1.89	205,164	2.52	40,971
7	Seoul (S-Korea)	1.609	226	11,764	52.0	16,301	72.1	1394
8	Edmonton (Canada)	1.627	181,262	95,083	0.52	127,203	0.70	56,965
9	Calgary (Canada)	1.630	96,779	130,982	1.35	185,201	1.91	49,842
10	Amsterdam (Holland)	1.730	1152	62,630	54.4	88,977	77.2	8048
11	Brussels (Belgium)	1.745	943	54,941	58.3	75,833	80.4	7273
12	Delhi (India)	1.779	5078	198,972	39.2	270,302	53.2	27,063

Table 1. *Cont.*

ID	City (Country)	Fractal Dimension	Area (km^2)	OSM # Vertex	OSM ρ Vertex	OSM # Edge	OSM ρ Edge	$\sum$Edge (km)
13	Dallas (USA)	1.800	4688	116,034	24.7	166,882	35.6	26,376
14	Milan (Italy)	1.802	8043	265,604	33.0	356,506	44.3	36,666
15	Munich (Germany)	1.811	1817	188,387	103.7	253,015	139.2	17,380

6. Conclusions

The E-GTSP, which is an extension of the famous TSP, is proven by researchers to model a more realistic real-life combinatorial optimization problem, where the task is to find a close/open Hamiltonian cycle that visits exactly one vertex from each group in a given city. A recommended approach to solve this involves its reduction to the corresponding TSP before solving it to optimality. However, this transformation increases the dimension of the problem, thus making it computationally exhaustive (consuming time and RAM space) to solve for cases where the cost is a function of time. In this study, the authors have presented five different search algorithms for an E-GTSP to TSP transformation that operate spatially. They do not increase the count of the vertices or edges at any stage. They are tested on 15 selected cities, taken from the OSM dataset and classified into different road-networks, with five different $|V^s|$ instances each.

It is observed that, with increase in the complexity of a problem, all algorithms become erroneous, thereby making longer routes independent of the city's street-pattern (Figure 4a). For $|V^s| = 1$ instances, the D-Search is better among all discussed approaches; however, as one increases the value of $|V^s|$, i.e., instance complexity, R-Search gets better (Figure 4b). Figures 5b and 9 further support this observation by showing the minimum route-length value for R-Search. The performance of R-Search in this study could be attributed to its ability to consider stations belonging to regions lying outside the start–end proximity region (Figure 6). It has also been observed that with increase in the optimal route length, its corresponding length error from D-Search or R-Search also gets bigger irrespective of the complexity of the instance (Figure 7). One final observation, which proves the convoluted road networks of cities worldwide, is done in Figure 8, where a deviation of data points away from the mean value line towards some saturation value is observed. It shows that, for a higher number of groups, i.e., increased complexity, the optimal route length of different start–end pairs becomes almost similar in value irrespective of their spatial separation.

This study, thus, brings forth a search criterion of point/vertex selection from each group for an E-GTSP to TSP transformation by testing many real-road networks. This search criterion, i.e., R-Search, considers the radial/spatial spread of points from each group with respect to the start and end point for the best possible selection. Readers can download the used dataset/meta-data from [33]. A few other observations have also been documented regarding urban networks, which is believed to be helpful for urban planners.

Future research might involve the comparison of the R-Search criterion with other algorithms, like the state-of-the-art Lin–Kernighan–Helsgaun TSP solver, for E-GTSP to TSP conversion. Additional work might involve the development of a heuristic-R-Search (argued in this study) or ANN-R-Search algorithm. It is believed that this study will help geospatial and urban developers to solve location based problems that can be modelled into GTSP or E-GTSP by taking the spatial spread of points of interest into account.

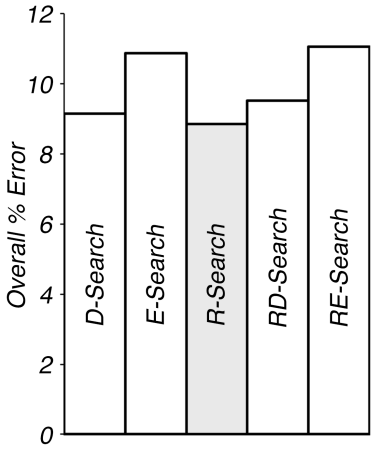

Figure 9. Percentage error of all proposed search algorithms. R-Search is better among all other searches. The authors do believe that this spatial approach for E-GTSP to TSP transformation could be improved by further incorporating heuristic concepts, which is beyond the scope of the current study.

Acknowledgments: The research presented in this article is primarily funded by The Scientific and Technological Research Council of Turkey under 2215—Graduate Scholarship Programme for International Students (TUBITAK, URL: https://www.tubitak.gov.tr/en). The authors would also like to thank the National Innovation and Research and Innovation Center for Geographical Information Technologies for additional support.

Author Contributions: Mohammed Zia and Ziyadin Cakir conceived and designed the experiments; Mohammed Zia performed the experiment; Mohammed Zia, Ziyadin Cakir, and Dursun Zafer Seker analyzed the data and made the conclusions; Mohammed Zia wrote the paper; Dursun Zafer Seker improved the text.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

E-GTSP	Equality-Generalized Traveling Salesman Problem
GTSP	Generalized Traveling Salesman Problem
TSP	Traveling Salesman Problem
OSM	OpenStreetMap
GTSPLIB	Generalized Traveling Salesman Problem Library
E-Search	Euclidean Search
D-Search	Dijkstra Search
R-Search	Radial Search
RE-Search	Radial Euclidean Search
RD-Search	Radial Dijkstra Search

References

1. Hamilton, W.R. Memorandum respecting a new system of roots of unity. *Philos. Mag.* **1856**, *12*, 446.
2. Lenstra, J.K.; Kan, A.H.G.R. Some Simple Applications of the Travelling Salesman Problem. *Oper. Res. Quart.* **1975**, *26*, 717–733, doi:10.1057/jors.1975.151.
3. Psaraftis, H.N.; Wen, M.; Kontovas, C.A. Dynamic vehicle routing problems: Three decades and counting. *Networks* **2016**, *67*, 3–31, doi:10.1002/net.21628.
4. Henry-Labor, A.L. The record balancing problem: A dynamic programming solution of a genera salesman problem. *RAIRO-Oper. Res.* **1969**, *2*, 43–49.
5. Saksena, J.P. Mathematical model for scheduling clients through welfare agencies. *Can. Oper. Res. Soc.* **1970**, *8*, 185–200.
6. Srivastava, S.S.; Kumar, S.; Garg, R.C.; Sen, P. Generalized travelling salesman problem through n sets of nodes. *Can. Oper. Res. Soc.* **1969**, *7*, 97–101.

7. Laporte, G.; Nobert, Y. Generalized Travelling Salesman Problem Through n Sets Of Nodes: An Integer Programming Approach. *INFOR Inf. Syst. Oper. Res.* **1983**, *21*, 61–75, doi:10.1080/03155986.1983.11731885.
8. Noon, C.E.; Bean, J.C. A Lagrangian Based Approach for the Asymmetric Generalized Traveling Salesman Problem. *Int. J. Prod. Res.* **1991**, *39*, 623–632.
9. Helsgaun, K. Solving the equality generalized traveling salesman problem using the Lin–Kernighan–Helsgaun Algorithm. *Math. Program. Comput.* **2015**, *7*, 269–287, doi:10.1007/s12532-015-0080-8.
10. Karp, R.M. Reducibility among Combinatorial Problems. In *50 Years of Integer Programming 1958–2008*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 219–241; doi:10.1007/978-3-540-68279-0_8.
11. Laporte, G.; Asef-Vaziri, A.; Sriskandarajah, C. Some Applications of the Generalized Travelling Salesman Problem. *J. Oper. Res. Soc.* **1996**, *47*, 1461–1467, doi:10.1057/jors.1996.190.
12. Ben-Arieh, D.; Gutin, G.; Penn, M.; Yeo, A.; Zverovitch, A. Process planning for rotational parts using the generalized travelling salesman problem. *Int. J. Prod. Res.* **2010**, *41*, 2581–2596, doi:10.1080/0020754031000087337.
13. Gutin, G., Punnen, A. P. The Traveling Salesman Problem and Its Variations. *Comb. Optim.* **2007**, *12*, 830, doi:10.1007/b101971.
14. Helsgaun, K. General k-opt submoves for the Lin–Kernighan TSP heuristic. *Math. Program. Comput.* **2009**, *1*, 119–163, doi:10.1007/s12532-009-0004-6.
15. Lawler, E.L.; Lenstra, J.K.; Kan, A.H.G.R.; Shmoys, D.B. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*; Wiley: Hoboken, NJ, USA, 1985; p. 476, ISBN 978-0-471-90413-7.
16. Fu, C.; Cai, D. EFANNA: An Extremely Fast Approximate Nearest Neighbor Search Algorithm Based on kNN Graph. *arXiv* **2016**, arXiv:1609.07228.
17. Okabe, A.; Boots, B.; Sugihara, K.; Chiu, S.N.; Kendall, D.G. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*; Wiley: Hoboken, NJ, USA, 2008; doi:10.1002/9780470317013.
18. Zia, M.; Cakir, Z.; Seker, D.Z. A New Spatial Approach for Efficient Transformation of Equality—Generalized TSP to TSP. *Free Open Source Softw. Geospat. (FOSS4G) Conf. Proc.* **2017**, *17*, 15-22, doi:10.7275/R53B5XBG.
19. Laporte, G.; Mercure, H. Generalized travelling salesman problem through n sets of nodes: The asymmetrical case. *Discret. Appl. Math.* **1987**, *18*, 185–197, doi:10.1016/0166-218X(87)90020-5.
20. Ben-Arieh, D.; Gutin, G.; Penn, M.; Yeo, A.; Zverovitch, A. Transformations of generalized ATSP into ATSP. *Oper. Res. Lett.* **2003**, *31*, 357–365, doi:10.1016/S0167-6377(03)00031-2.
21. Dimitrijevic, V.; Saric, Z. An efficient transformation of the generalized traveling salesman problem into the traveling salesman problem on digraphs. *Inf. Sci.* **1997**, *102*, 105–110, doi:10.1016/S0020-0255(96)00084-9.
22. Laporte, G.; Semet, F. Computational Evaluation Of A Transformation Procedure For The Symmetric Generalized Traveling Salesman Problem. *INFOR Inf. Syst. Oper. Res.* **2016**, *37*, 114–120, doi:10.1080/03155986.1999.11732374.
23. Lien, Y.; Ma, E.; Wah, B.W.S. Transformation of the generalized traveling-salesman problem into the standard traveling-salesman problem. *Inf. Sci.* **1993**, *74*, 177–189, doi:10.1016/0020-0255(93)90133-7.
24. Noon, C.E.; Bean, J.C. An Efficient Transformation Of The Generalized Traveling Salesman Problem. *INFOR Inf. Syst. Oper. Res.* **1993**, *31*, 39–44, doi:10.1080/03155986.1993.11732212.
25. Bellman, R. Dynamic Programming Treatment of the Travelling Salesman Problem. *J. ACM* **1962**, *9*, 61–63, doi:10.1145/321105.321111.
26. Fischetti, M.; Salazar, J.J.; Toth, P. A Branch-and-Cut Algorithm for the Symmetric Generalized Traveling Salesman Problem. *Oper. Res. Lett.* **1997**, *45*, 378–394, doi:10.1287/opre.45.3.378.
27. Kirkpatrick, S.; Gelatt, C.D., Jr.; Vecchi, M.P. Optimization by Simulated Annealing. *Science* **1983**, *220*, 671–680, doi:10.1126/science.220.4598.671.
28. Munson, D. Which Street Pattern Represents Your Continent? 2013. Available online: https://munsonscity.com/2013/10/09 (accessed on 18 January 2018).
29. Bourke, P. Fractal Dimension Calculator—FDC. 2003. Available online: http://paulbourke.net/fractals/fracdim (accessed on 18 January 2018).
30. Overpass API. OpenStreetMap Overpass API. 2016. Available online: http://overpass-api.de (accessed on 18 January 2018).
31. Hijmans, R.; Kapoor, J.; Wieczorek, J.; Garcia, N.; Maunahan, A.; Rala, A.; Mandel, A. GADM Database of Global Administrative Areas. 2015. Available online: http://www.gadm.org (accessed on 18 January 2018).

32. Zverovitch, A. GTSP iNstances. 2008; Volume 2002. Available online: http://www.cs.rhul.ac.uk/home/zvero/GTSPLIB (accessed on 18 January 2018).
33. Zia, M. E-GTSP-to-TSP-Tested-Instances. 2016. Available online: https://github.com/Zia-/E-GTSP-to-TSP-tested-instances (accessed on 18 January 2018).