https://doi.org/10.1016/j.cor.2016.09.003

A Novel Discretization Scheme for the Close Enough Traveling Salesman Problem

Francesco Carrabs[1,*], Carmine Cerrone[3], Raffaele Cerulli[1], Manlio Gaudioso[2]

Abstract

This paper addresses a variant of the Euclidean traveling salesman problem in which the traveler visits a node if it passes through the neighborhood set of that node. The problem is known as the close-enough traveling salesman problem. We introduce a new effective discretization scheme that allows us to compute both a lower and an upper bound for the optimal solution. Moreover, we apply a graph reduction algorithm that significantly reduces the problem size and speeds up computation of the bounds. We evaluate the effectiveness and the performance of our approach on several benchmark instances. The computational results show that our algorithm is faster than the other algorithms available in the literature and that the bounds it provides are almost always more accurate.

Keywords: Close-Enough, Traveling Salesman Problem, Discretization scheme

1. Introduction

This paper concerns the close-enough variant CETSP of the classic traveling salesman problem (TSP). Given a set of target points in any Euclidean space, the TSP consists in determining a minimum length tour that starts and ends at a "depot" while visiting each target point exactly once. In CETSP to each

*Corresponding author

Email addresses: `fcarrabs@unisa.it` (Francesco Carrabs), `carmine.cerrone@unimol.it` (Carmine Cerrone), `raffaele@unisa.it` (Raffaele Cerulli), `manlio.gaudioso@unical.it` (Manlio Gaudioso)

[1]Department of Mathematics, University of Salerno, Italy.

[2]Department of Mechanical, Energy and Management Engineering - University of Calabria, Italy.

[3]Department of Biosciences and Territory, University of Molise, Italy.

target point v is associated a neighborhood, that is a compact region of the space containing v. In fact CETSP consist in finding the shortest tour that starts and ends at the depot and intersects each neighborhood once. Usually, there are no constraints on the shape of the neighborhood, on the other hand the disc shape is the mostly adopted one, thus we keep this assumption in our paper.

The CETSP has a number of practical applications. For instance, let us consider a district where a certain number of radio frequency identification (RFID) readers are located to record electricity or water or gas consumption. Each meter reader plays the role of a target point and its information can be relayed within a fixed range r. Consequently, the neighborhood is defined as a disc of radius r centered at each target point. The reading process of the RFID reader meters can be performed by flying a drone within the neighborhood of each target point, speeding up the classic door-to-door reading. Other applications of CETSP arise in the robot monitoring of wireless sensor networks [13] and in the context of Unmanned Aerial Vehicles for aerial forest fire detection or military surveillance.

The CETSP was introduced by Gulczynski et al. [7] and the authors proposed six heuristics to solve the problem under the assumption that all neighborhoods were discs of the same radius. For the same problem, Dong et al. [4] introduced two heuristics based on the concept of supernodes. A supernode S is a set of points of the plane such that for each target point v there exists at least one point in S whose distance from v is at most r, the radius of the disc. Supernodes are generated by using convex hull and clustering techniques. A mixed integer nonlinear programming formulation of CETSP was provided too, but it was not specifically used in algorithm design. Mennell et al. [10, 11] proposed another heuristic based on the intersection of the neighborhoods, named Steiner zone. Yuan et al. [13] developed an effective evolutionary approach which was able to find the shortest tour on all the benchmark instances, although with large computation time. Other heuristics were proposed by Shuttleworth et al. [12] to solve the CETSP over a street network for the specific RFID meter

reader application described above. Finally, some special cases of CETSP were solved by polynomial-time approximation algorithms introduced by Arkin and Hassin [1], Mata and Mitchell [9] and Dumitrescu and Mitchell [5].

In this paper we introduce an approach to compute upper and lower bounds for the CETSP problem by discretizing the solution space and solving on the resulting graph the classic Generalized TSP problem. Starting from the discretization approaches already proposed in literature, we introduce a new effective discretization scheme which provides better bounds, thanks to a novel adaptive approach to select appropriately the number of discretization points to be used for each neighborhood. Finally, we apply a graph reduction algorithm that significantly reduces the size of the Generalized TSP problem to solve, allowing us to increase the number of discretization points without penalizing the performance of our approach in terms of CPU time. The computational results, carried out on benchmark instances, reveal that our approach outperforms the ones proposed in [2].

The remainder of the paper is organized as follows. Section 2 introduces the definitions and the notations that are used throughout the paper. Section 3 and Section 5 present our discretization scheme and our graph reduction algorithm, respectively. The mixed-integer programming model (MIP) is described in Section 6 and it is followed by the computational results in Section 7. Finally, conclusions are presented in Section 8.

2. Definitions and Notation

Let N be a set of points in a two-dimensional plane, with $|N| = n$, and let p_0 be the depot point. We will refer to the elements in N as the *target points*. To each target point v is associated a sphere C_v with center v and radius r_v (Figure 1(a)) which will be referred to as the *neighborhood* $N(v)$ of v. W.l.o.g., we suppose that $p_0 \notin N(v) \, \forall v \in N$. The CETSP consists in finding a shortest tour T^* that starts from the depot p_0, intersects every neighborhood $N(v)$ (in any order), and ends in p_0. The points of any tour T where a direction change

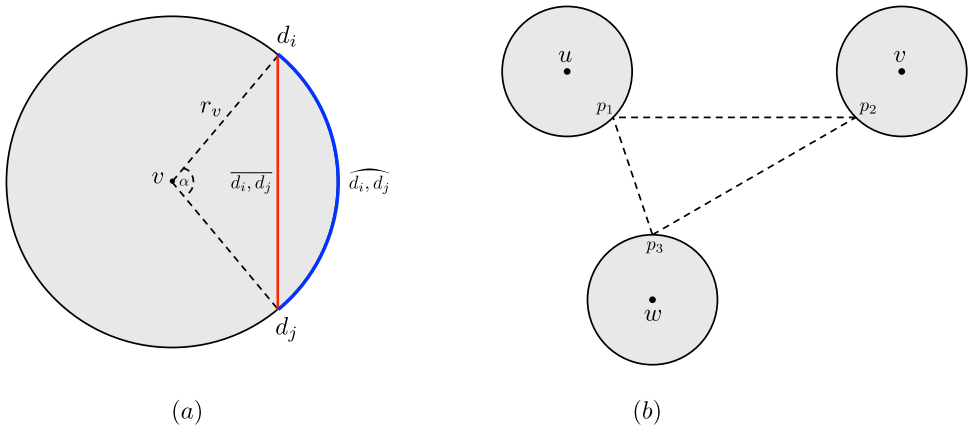

Figure 1: (a) Neighborhood $N(v)$ of the target point v. (b) A feasible tour for the CETSP problem.

occurs are the *turn points* and any tour can be uniquely identified through its turn points. For instance, in Figure 1(b) it is shown a feasible tour T for the CETSP which is identified by the turn points p_1, p_2 and p_3. Given a couple of turn points p_i and p_j, the length of the edge (p_i, p_j) is given by the euclidean distance between p_i and p_j. The total cost of a tour T is denoted by $w(T)$ and it is equal to the sum of the edge lengths in T.

Given the neighborhood $N(v)$ depicted in Figure 1(a), let d_i and d_j be two points of the boundary of C_v. We denote by $\overline{d_i, d_j}$ the *chord* between these two points and by $\widehat{d_i, d_j}$ the *circular arc* from d_i to d_j in the clockwise direction. For any additional definition and notation on the graphs we refer to [3].

3. The Perimetral Discretization scheme

Since each neighborhood $N(v)$, $v \in N$, contains an infinite number of turn points, then the number of feasible tours for CETSP is infinite as well. However, a finite number of turn points occur for any feasible tour. For this reason, we can associate to each feasible solution a discrete set of points. More in details, each neighborhood $N(v)$ is discretized by using a fixed number k of discretization points. We denote by $\hat{N}(v)$ such set of points. Consequently, a graph $G =$

(V, E), where $V = \bigcup_{v \in N} \hat{N}(v)$ and $E = \{(x, y) : x \in N(u), y \in N(v), u \neq v\}$, is build. It is easy to see that the weight of any tour $\hat{T}$, that starts and ends at the depot and that visits exactly one discretization point in each neighborhood, is an upper bound to $w(T^*)$. From now on, we will denote by T and $\hat{T}$ the feasible tours of the CETSP computed by using the points of $N(v)$ and of $\hat{N}(v)$, $v \in N$, respectively.

In order to have an upper bound of $w(T^*)$ as tight as possible, we compute the shortest tour $\hat{T}^*$ in G, namely we solve the *Generalized TSP* problem (GTSP) on G. The quality of the bound $\hat{T}^*$ for $w(T^*)$ heavily depends on the number of points used to carry out the discretization and on their placement in each neighborhood. Obviously greater the number of discretization points tighter will be $w(\hat{T}^*)$, but, on the other hand, greater will be the size of G with increasing computational cost to calculate $\hat{T}^*$. For this reason it is necessary to find an appropriate trade-off between the quality of the upper bound and the time spent to compute it. Moreover, it is crucial to use a discretization scheme that minimizes the discretization error carried out in each neighborhood due to the use of discretization points. More in details, let us consider the example in Figure 2 where $N = \{v_1, v_2, v_3\}$ and T^* is the optimal tour for the CETSP, identified by the turn points p_1, p_2, p_3 and the depot p_0. Note that the turn points of T^* are always on the boundary of the spheres (see Proposition 1 in the sequel). Each neighborhood is discretized by using only $k = 2$ discretization points placed on the corresponding circumference.

Let us build now the *walk* $Q = \{p_0, p_1, d_1, p_1, p_2, d_2, p_2, p_3, p_0\}$. In practice Q is built by following the edges of T^* and, for each turn point $p_i \in C_{v_i}$, the closest discretization point $d_i \in \hat{N}(v_i)$ is detected and the chord $\overline{p_i, d_i}$ is crossed twice. We define the *discretization error* $\xi(v_i)$ as two times the length of $\overline{p_i, d_i}$. Thus $\xi(v_i)$ represents the error in $\hat{N}(v_i)$ with respect to T^*, due to the choice of the discretization points. It is easy to see that:

$$w(Q) = w(T^*) + \sum_{v \in N} \xi(v) \tag{1}$$

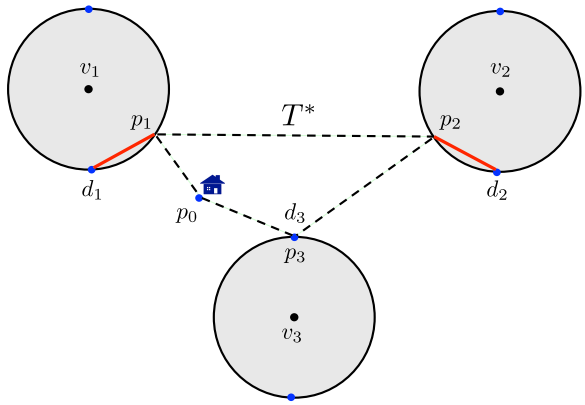

Figure 2: The discretization error carried out in each neighborhood.

Since Q starts and ends at the depot p_0 and visits one discretization point for each neighborhood, then $w(\hat{T}^*) \leq w(Q)$. This means that the lower is the discretization error carried out in each neighborhood, the tighter will be $w(\hat{T}^*)$. For this reason, it is very important to apply a discretization scheme that minimizes $\sum_{v \in N} \xi(v)$.

The following proposition ([2]) suggests a possible discretization scheme.

Proposition 1. *Any optimal solution T^* to the CETSP can be represented by a finite set of connected line segments (p_0, p_1), (p_1, p_2), (p_{k-1}, p_k), (p_k, p_0), where $k \leq n$, and for each point p_i, $i = 1 \ldots k$, there exists at least one $v \in N$ such that p_i is on the circumference C_v.*

From Proposition 1, a quite intuitive discretization scheme consists in placing the discretization points on the circumferences associated to the target points. Such a scheme is named *perimetral discretization* (PD) scheme. More in details, let k be the number of points used to discretize each neighborhood $N(v)$. Then the PD scheme divides each circumference C_v in k equal circular arcs and places the discretization points at the extremes of these circular arcs. Let α be the angle associated to any circular arc $\widehat{d_i, d_j}$ that is $\alpha = \frac{2\pi}{k}$. The Figure 3(a) shows a PD scheme for $k = 3$, where $\alpha = 120°$ and $\hat{N}(v) = \{d_1, d_2, d_3\}$.

It is necessary now to evaluate the maximal discretization error associated to

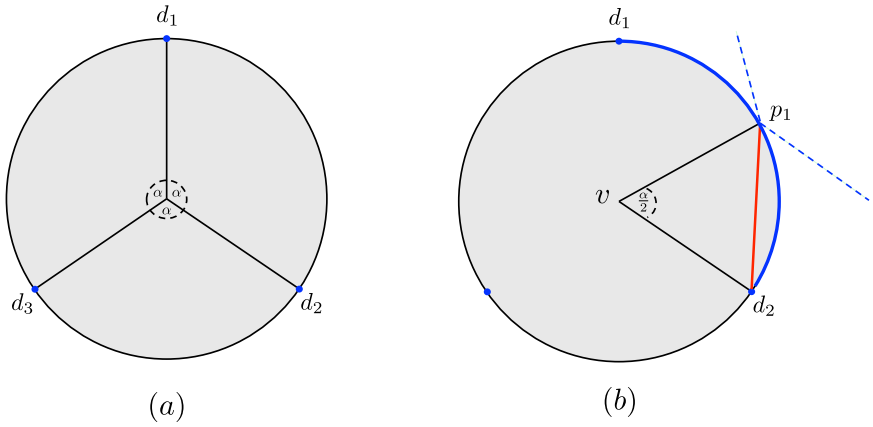

Figure 3: (a) Perimetral discretization for $k = 3$ with $N(v) = \{d_1, d_2, d_3\}$ and $\alpha = 120°$. (b) The maximum discretization error for the PD scheme.

PD scheme. Given a neighborhood $\hat{N}(v)$, the worst case occurs when the turn point p_1 of T^* touches the circumference C_v in the middle of the circular arc $\widehat{d_1, d_2}$ (Figure 3(b)). Indeed, in this case we have the maximum distance between p_1 and the closest discretization point d_1 (or d_2). We have $w(\overline{p_1, d_2}) = 2r_v \sin(\frac{\alpha}{4})$ and then $\xi(v) = 4r_v \sin(\frac{\alpha}{4})$. In general, once the number of discretization points k is fixed, the maximum error for the PD scheme in $\hat{N}(v)$ is $\xi(v) = 4r_v \sin(\frac{\pi}{2k})$.

4. The Internal Discretization scheme

In the previous section we described an intuitive and widely used discretization scheme for the CETSP problem. In this section we prove that it is possible to reduce the discretization error with respect to PD scheme, by placing the discretization points inside the neighborhoods rather than on the circumferences as suggested by the following Proposition 1. This new scheme is named *internal point discretization* (IP) and works as follows. Given $k$the number of discretization points, the IP scheme divides C_v in k equal circular arcs and, for each arc $\widehat{a, b}$, places a discretization point in the middle of the chord $\overline{a, b}$. In Figure 4(a) the IP schemes are shown for $k = 3$ and $k = 4$.

Application of the IP scheme assures that whenever the touch point p_1 of T^* is on the circular arc $\widehat{a, b}$, the maximum distance between p_1 and the closest

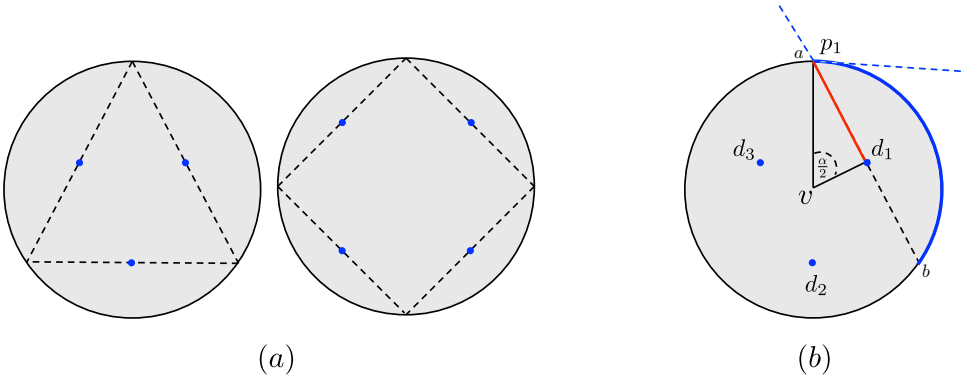

Figure 4: (a) The internal point discretization scheme for $k = 3$ and $k = 4$. (b) The maximum error in the IP scheme.

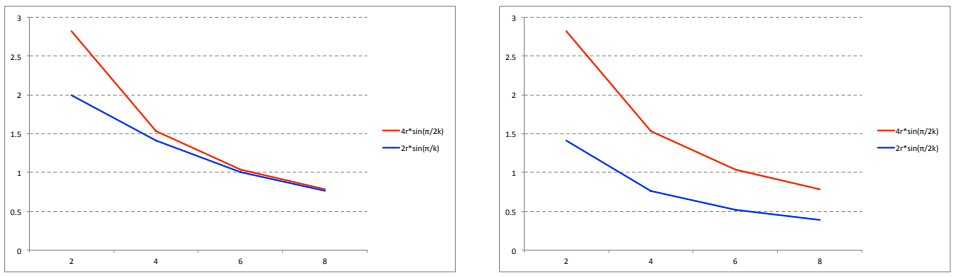

Figure 5: (a) The maximum discretization error of PD (red) and IP (blue) schemes. (b) The maximum discretization error for AIP and ADS on a neighborhood intersecting the boundary of the convex hull.

discretization point of $\hat{N}(v)$ is equal to half the length of the $w(\overline{a,b})$ segment (Figure 4(b)). It is $w(\overline{a,b}) = r_v \sin(\frac{\alpha}{2})$, where α is the angle associated to the circular arc $\widehat{a,b}$. As consequence, for fixed k, the maximum error associated to the IP scheme in $\hat{N}(v)$ is equal to $\xi(v) = 2r_v \sin(\frac{\pi}{k})$.

We compared the discretization errors of PD and IP schemes for radius equal to 1 and k equal to 2, 4, 6 and 8. The result of this comparison is depicted in Figure 5(a).

Note that the discretization error of IP scheme is always lower than the one of PD scheme. In particular, the smaller is k the greater is the gap between the two discretization errors. For instance, when $k = 2$ the gap between the

two errors is equal to 0.8. This means that, in the worst case, our solution is $0.8 \times n$ units lower than the solution produced by PD scheme with $k = 2$. By increasing the value of k the gap decreases and for $k = 8$ it is fairly small. In [2] only 4 discretization points are used to discretize each neighborhood due to performance problem. This means that, in the worst case, a discretization error equal to 1.5 occurs in each neighborhood. Instead, our approach is so fast that we can use 8 discretization points for each neighborhood without penalizing the performance. As a consequence, in the worst case we pay an error equal to 0.76, instead that 1.5, as shown in Figure 5(a).

In next section we will describe a methodology to further reduce the discretization error of IP scheme which will allow us to get better upper bounds.

4.1. Convex Hull Strategy

A significant result about discretization of CETSP problem is the following proposition proven in [2]:

Proposition 2. *Let T be an optimal CETSP tour that is characterized by a set of turn points $p_0, \cdots, p_k$. Then $p_i \in conv(N \cup \{p_0\})$, for $i = 1, \cdots, k$, where conv denotes the convex hull of any set of points.*

This proposition states that all points that are outside $conv(N \cup \{p_0\})$ can be discarded because they cannot belong to the optimal solution. In other words, the discretization points should be used only to discretize the circular arcs within the convex hull rather than the whole circumference. From now on the circular arcs within the convex hull are referred to as *feasible circular arcs* (*fca*). Moreover, given a target point v_i, we denote by $\widehat{v_i', v_i''}$ the *fca* of C_{v_i} and by α_i the central angle associated to $\widehat{v_i', v_i''}$. In [2] such idea is implemented by discretizing the *fca* but only for the neighborhoods located at the corners of the convex hull. In Figure 6(a) this type of discretization named *arc discretization scheme* (ADS) is depicted.

The discretization points are placed on the *fca* (in blue) of the circumferences C_{v_1},C_{v_2},C_{v_4},C_{v_5} and C_{v_7}. Instead, for the neighborhoods $\hat{N}(v_3)$ and $\hat{N}(v_6)$ the

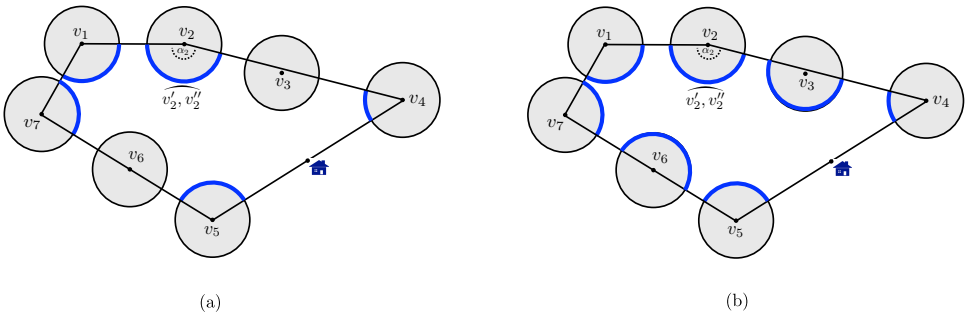

Figure 6: (a) The arc discretization scheme. (b) Our adaptive internal point scheme.

discretization points are placed on whole circumferences C_{v_3} and C_{v_6} because they are not at the corners of the convex hull.

By using Proposition 2, we introduce a new discretization scheme, which is referred to as *internal arc discretization* (AIP) scheme. It is more effective than IP and can be obtained on the basis of the following two ideas.

The first one consists in discretizing only the *fca* for all the neighborhoods, with no use of discretization points for circular arcs which are outside the convex hull. To this end, it is necessary to detect any neighborhood $N(v)$ that intersects a boundary of the convex hull at two points because, under this condition, $\widehat{v', v''}$ is smaller than C_v. For instance, in Figure 6(b) the AIP will discretize the two *fca* $\widehat{v_3', v_3''}$ and $\widehat{v_6', v_6''}$ rather than the whole circumferences as carried out by the ADS scheme. As a consequence, for the same k, the discretization error made by AIP on $\hat{N}(v_3)$ and $\hat{N}(v_6)$ is lower than the discretization error made by the ADS scheme because for the AIP scheme the value of α is smaller. More in details, let us consider the circumference C_{v_6} whose center is on the boundary of the convex hull (Figure 6(b)). By applying the AIP scheme we discretize the *fca* $\widehat{v_6', v_6''}$ and then $\alpha_6 = \pi$ and the discretization error is $\xi(v_6) = 2r_{v_6} \sin(\frac{\pi}{2k})$. Instead, the ADS scheme discretizes the whole C_{v6} and then $\alpha_6 = 2\pi$ and $\xi(v_6) = 2r_{v_6} \sin(\frac{\pi}{k})$. Figure 5(b) shows the discretization error associated to the two schemes for neighborhood $N(v_6)$. The results for the ADS are the same as PD while the discretization error for AIP is significantly lower than IP.

Second idea comes from the observation (see Figure 5(b)) that the length of

the various *fca* can vary largely. Thus, rather than using the same number k of points for each *fca* we introduce an *adaptive* approach fixing the number of discretization points to be used for each *fca* according to its length. Note that, for instance, the approaches currently adopted use k points to discretize both $\widehat{v'_6, v''_6}$ and $\widehat{v'_4, v''_4}$ (Figure 6(b)) even if their lengths are quite different. In fact, in terms of error, k discretization points would be too many for $\widehat{v'_4, v''_4}$ and too few for $\widehat{v'_6, v''_6}$.

Summing up, given a neighborhood $N(v_i)$, we apply an adaptive approach that selects the number of discretization points according to the degree range of α_i. More in details, let $\hat{\alpha}$ be the *degree step* given by the ratio between the sum of all central angles α_i and the total number of discretization points $k|N|$. Formally, $\hat{\alpha} = \dfrac{\sum_{v_i \in N} \alpha_i}{k|N|}$. Then, the *fca* $\widehat{v'_i, v''_i}$ is discretized by using $\frac{\alpha_i}{\hat{\alpha}}$ points. According to the two improvements described in this section, the new discretization error $\xi(v_i)$, carried out by AIP on the neighborhood $\hat{N}(v_i)$, is expressed as: $\xi(v_i) = 2r_{v_i} \sin(\frac{\alpha_i}{2k_i}) = 2r_{v_i} \sin(\frac{\hat{\alpha}}{2})$ where $k_i = \frac{\alpha_i}{\hat{\alpha}}$.

5. Graph Reduction Algorithm (GRA)

In previous section 4 we have described how to compute an upper bound for $w(T^*)$ by finding the shortest tour $\hat{T}^*$ on the graph $G(V, E)$. The size of G depends on the number of target points n and the number of discretization points k used (it is $|V| = nk$ and $|E| = \frac{k^2 n(n-1)}{2}$). The selection of an appropriate value for k is crucial for both the performance and the effectiveness of the MIP model because for high values of k the upper bound we obtain is tighter but calculation of $\hat{T}^*$ is expensive, while, for low values of k, $\hat{T}^*$ can be found quickly, at the expenses of an usually poor upper bound.

In this section we introduce an algorithm which detects the *useless edges* of G, that is those edges that cannot belong to the optimal solution $\hat{T}^*$. Thanks to this algorithm the size of G is significantly reduced, allowing us to use the double of the discretization points adopted by [2], which results in a beneficial effect on the quality of the solution, without impairing performance in terms of

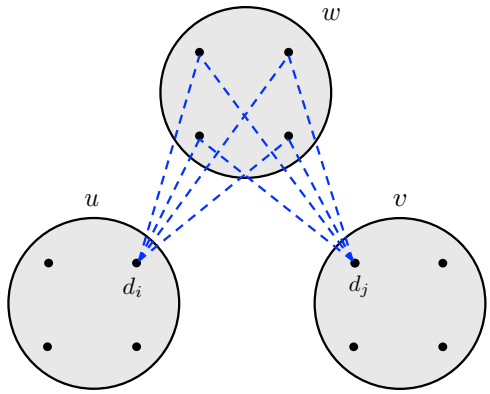

Figure 7: The idea behind the graph reduction algorithm.

computational time.

The idea of applying a preprocessing phase on a graph G before solving the GTSP problem has been already introduced in [8]. However, the idea behind our *Graph Reduction Algorithm* (GRA), described in the following, allows an easier implementation with respect to the algorithm proposed in [8], because it not requires sorting operations.

GRA works as follows. Given a target point $w \in N$, let d_i and d_j be two discretization points such that $d_i \in \hat{N}(u)$, $d_j \in \hat{N}(v)$ and $u \neq v \neq w$ (Figure 7). According to the euclidean distance, the GRA computes the shortest path between d_i and d_j, passing through d, for each point $d \in \hat{N}(w)$, and it marks as needed the two edges of the shortest path. The algorithm repeats this operation for each target point $w \in N$ and for all possible couples d_i and d_j. At the end of the computation, GRA removes all not-marked edges. Proposition 3 ensures that the edges removed by GRA cannot belong to $\hat{T}^*$.

Proposition 3. *Given a target point $w \in N$, let S be the set of shortest paths from $d_i \in \hat{N}(u)$ to $d_j \in \hat{N}(v)$ passing through any point $d \in \hat{N}(w)$, with $u \neq v \neq w$. Then any edge $(d_x, d_y) \in E$, incident to $\hat{N}(w)$ and outside all shortest path of S, cannot belong to the optimal solution $\hat{T}^*$.*

Proof. The proof is by contradiction. Suppose that (d_x, d_y) belongs to $\hat{T}^*$ and w.l.o.g. let $d_x \in \hat{N}(w)$ and d_r be the discretization point coming just before

d_x in $\hat{T}^*$. Since (d_x, d_y) does not belong to any shortest path in S, then there exists another discretization point $d'_x \in \hat{N}(w)$ such that the path $\{d_r, d'_x, d_y\}$ is shorter than $\{d_r, d_x, d_y\}$. By replacing $\{d_r, d_x, d_y\}$ with $\{d_r, d'_x, d_y\}$ in $\hat{T}^*$, we obtain a discretized tour better than the optimal one. A contradiction. $\square$

The application of the GRA algorithm has a tremendous impact on the performance of our MIP model because $\hat{T}^*$ is computed on a reduced graph G' with a fairly smaller set of edges with respect to G. Indeed, during our computational tests we observed reductions up to 80% of $|E|$.

As far as complexity is concerned, observe that we define first $\mathcal{N} = nk$, the total number of discretization points in G. For any fixed target point w and a couple of discretization points d_i and d_j, the computation of the shortest path from d_i to d_j passing through a discretization point of $\hat{N}(w)$ can be carried out in $O(k)$, as each possible path is composed by only two edges and then it is sufficient to compare the cost of the k possible paths to detect the shortest one. Now, this operation is repeated for each discretization point $w \in N$, that is n times, and for all the possible couples of discretization points $d_i \in \hat{N}(u)$ and $d_j \in \hat{N}(v)$ such that $u \neq v \neq w$. An upper bound on the number of such couples is $\binom{\mathcal{N}}{2} = O(\mathcal{N}^2)$. As a consequence, the running time of GRA is $O(\mathcal{N}^3)$.

6. Mathematical Formulation

The input of the MIP model is the reduced graph $G'(V, E')$ obtained by applying the GRA algorithm to $G(V, E)$. To formulate the GTSP problem, we associate to each edge $(i, j) \in E'$ a binary variable x_{ij} taking value 1 if and only if (i, j) belongs to the solution. Moreover, we associate to each discretization node i the binary variable y_i taking value 1 if and only if i belongs to the solution. Finally, we let c_{ij} be the euclidean distance between the discretization points i and j and define the set $E(S)$ as follows:

$$E(S) = \{(i, j) \in E' : i, j \in \bigcup_{v \in S} \hat{N}(v)\}$$

for $S \subseteq N$. Our integer linear programming model for the GTSP is the following:

$$
\begin{aligned}
(\mathbf{MIP}) \qquad & \min \sum_{(i,j)\in E'} c_{ij} x_{ij} & & (2)\\
& \sum_{i\in \hat{N}(v)} y_i = 1 & \forall v \in N \quad & (3)\\
& \sum_{i\in \hat{N}(u), j\in \hat{N}(v)} x_{ij} \leq 1 & u, v \in N, u \neq v \quad & (4)\\
& \sum_{(i,j)\in E'} x_{ij} = 2y_i & \forall i \in V \quad & (5)\\
& \sum_{(i,j)\in E(S)} x_{ij} \leq |S| - 1 & \forall S \subseteq N, S \neq \emptyset \quad & (6)
\end{aligned}
$$

The objective function (2) minimizes the cost of the tour. Constraints (4) ensure that at most one edge connecting two neighborhoods is selected. Constraints (5) bind the two sets of variables by letting y_i equal to 1 if and only if v_i belongs to the solution. Finally, constraints (6) are the subtour elimination constraints adapted to the Generalized TSP [6].

The MIP model returns the optimal tour $\hat{T}^*$ with $w(\hat{T}^*)$ being our upper bound to the optimal solution T^* of CETSP. A lower bound for T^* can be found too by removing from $w(\hat{T}^*)$ the maximum discretization error value $\xi(v)$ for each target point v. Formally, $LB = w(\hat{T}^*) - \sum_{v\in V} \xi(v)$. Note that lower bound calculation is useful in view of comparison with other algorithms on the basis of the gap between upper and lower bound.

Finally, once the solution $\hat{T}^*$ has been found, we carry out an additional step to possibly improve the upper bound.

In fact the solution $\hat{T}^*$ can be improved by changing the position of the discretization points used. More in details, the idea is to apply an *elastic force* on all the edges of $\hat{T}^*$, which attracts the discretization points. The two forces on the two edges incident to a discretization point v move this point along the bisector of the angle between the two edges towards the border of the neighborhoods. Obviously we forbid the crossing of these borders. As a result, a new tour cheaper than $\hat{T}^*$ is obtained. In the rest of the paper $\hat{T}^*$ denotes such final

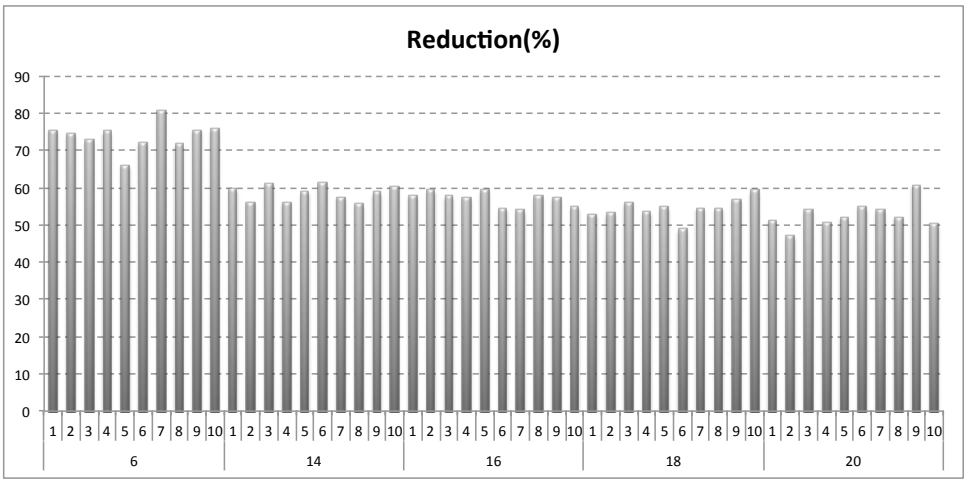

Figure 8: Arc set reduction obtained by applying the GRA algorithm.

tour.

7. Computational Results

This section presents the results of our computational test phase on the benchmark instances proposed in [2], where the authors introduce three approaches: a two-stage MIP formulation (LB3), a Bender Decomposition (BD) and an iterative Algorithm (IA), that are used to solve the various scenarios proposed. On the contrary, we use for all the scenarios a single approach, named ULB, consisting of *i*) applying the AIP scheme to discretize the neighborhoods, *ii*) applying the GRA algorithm to reduce the size of the graph G', *iii*) solving the GTSP problem on the graph G', by using the MIP model and *iv*) applying the elastic force algorithm to improve the upper bound. ULB was coded in Java on a OSX platform running on an Intel Core i5 2.9GHz processor with 16GB RAM, equipped with the IBM ILOG CPLEX 12.5.1 solver and the Concert Technology Library for the mathematical formulations. We start our computational study by evaluating the effectiveness of GRA algorithm.

In Figure 8 the percentage of edges removed from GRA algorithm is shown. The five scenarios, with 6, 14, 16, 18 and 20 target points, are reported on the

x-axis, each one composed by ten instances. It is interesting to observe that almost always more than 50% of edges are removed from GRA. In the worst case the reduction is around 45% but in the best case it reaches the 80%. Since the reduced graph G' contains a much smaller number of edges, the MIP model requires less computational time to find the optimal tour $\hat{T}^*$. This allows us to use $k = 8$ discretization points for each neighborhood, instead of the 4 used by Behdani and Smith, without significantly penalizing the performance of the algorithm ULB in terms of Cpu time. On the other hand, by using a higher value of k we expect to obtain more accurate bounds.

Before carrying out the comparison between ULB and the other algorithms proposed in literature, it is interesting to investigate the contribution to the ULB effectiveness given by the Convex Hull strategy and by the Elastic Force algorithm. As explained in section 4.1, the Convex Hull strategy allows to reduce the discretization error carried out at each neighborhood. Since our lower bound is computed according to this value, the lower the discretization error, the better is the computed lower bound. Accordingly, we expect an improvement of the lower bounds by applying the Convex Hull strategy. Instead, the Elastic Force algorithm is designed to improve only the upper bound computed by the MIP model and then it does not affect the lower bound. In order to verify the contribution of these two components, we implemented three variants of our algorithm. The *Basic* version is the ULB algorithm in which neither the Convex Hull strategy nor the Elastic Force algorithm are applied. The *Basic+CH* version is the basic version enriched by the Convex Hull strategy, while the *Basic+EL* one is the basic version plus the Elastic Force algorithm.

By comparing the upper and lower bounds computed by these three versions, we are able to quantify the effectiveness of the Convex Hull strategy and of the Elastic Force algorithm. Table 1 reports the results of this comparison. The computational tests are carried out on the largest instances proposed in [2] with a radius equal to 0.5. The first column of the table reports the scenario while the remaining six columns show the upper (UB) and lower (LB) bounds of Basic, Basic+CH and Basic+EL algorithms, respectively.

Scenario	Basic		Basic+CH		Basic+EL	
	UB	**LB**	**UB**	**LB**	**UB**	**LB**
CETSP-14-01	39.478	33.980	39.150	35.294	38.907	33.980
CETSP-14-02	37.490	31.992	37.143	33.199	36.934	31.992
CETSP-14-03	34.595	29.097	34.262	31.266	34.202	29.097
CETSP-14-04	38.552	33.054	38.225	34.551	37.945	33.054
CETSP-14-05	34.726	29.228	34.401	31.349	34.309	29.228
CETSP-14-06	34.612	29.114	34.355	30.731	34.164	29.114
CETSP-14-07	39.671	34.173	39.323	35.721	39.128	34.173
CETSP-14-08	35.172	29.674	34.851	31.159	34.590	29.674
CETSP-14-09	36.442	30.944	36.192	32.418	36.029	30.944
CETSP-14-10	35.415	29.917	35.228	31.383	34.984	29.917
CETSP-16-01	44.415	38.132	44.080	39.649	43.814	38.132
CETSP-16-02	35.959	29.676	35.615	31.174	35.356	29.676
CETSP-16-03	41.505	35.222	41.272	36.632	40.969	35.222
CETSP-16-04	37.071	30.788	36.780	32.562	36.537	30.788
CETSP-16-05	36.830	30.547	36.550	31.982	36.326	30.547
CETSP-16-06	41.260	34.977	40.832	36.851	40.626	34.977
CETSP-16-07	37.693	31.410	37.258	33.124	36.941	31.410
CETSP-16-08	36.857	30.574	36.556	32.467	36.340	30.574
CETSP-16-09	33.771	27.488	33.536	29.343	33.333	27.488
CETSP-16-10	35.211	28.928	34.862	30.660	34.641	28.928
CETSP-18-01	45.437	38.368	45.099	39.988	44.780	38.368
CETSP-18-02	38.781	31.712	38.363	33.497	38.078	31.712
CETSP-18-03	41.888	34.819	41.581	36.198	41.230	34.819
CETSP-18-04	42.432	35.363	41.996	37.159	41.763	35.363
CETSP-18-05	38.693	31.624	38.319	33.630	38.093	31.624
CETSP-18-06	37.818	30.749	37.497	32.550	37.224	30.749
CETSP-18-07	36.897	29.828	36.597	31.953	36.395	29.828
CETSP-18-08	34.084	27.015	33.836	29.250	33.712	27.015
CETSP-18-09	35.449	28.380	35.077	30.223	34.849	28.380
CETSP-18-10	42.227	35.158	41.873	36.806	41.546	35.158
CETSP-20-01	45.715	37.861	45.367	39.594	45.025	37.861
CETSP-20-02	36.684	28.830	36.192	31.199	35.925	28.830
CETSP-20-03	48.488	40.634	48.268	42.181	47.754	40.634
CETSP-20-04	39.491	31.637	39.166	33.328	38.803	31.637
CETSP-20-05	43.397	35.543	43.155	37.150	42.830	35.543
CETSP-20-06	44.142	36.288	43.760	38.095	43.403	36.288
CETSP-20-07	40.290	32.436	40.063	33.827	39.753	32.436
CETSP-20-08	39.036	31.182	38.772	33.110	38.487	31.182
CETSP-20-09	43.404	35.550	43.102	37.249	42.693	35.550
CETSP-20-10	43.099	35.245	42.827	36.903	42.302	35.245

Table 1: Impact evaluation of the Convex Hull strategy and Elastic algorithm on the effectiveness of our algorithm.

The upper bounds computed by Basic+CH are slightly better than the ones computed by Basic with a percentage gap that ranges from 0.45% (CETSP- 20-03) to 1.34% (CETSP-20-02). Instead, the Basic+CH lower bounds are much better than the Basic lower bounds with a percentage gap that ranges from 3.64% (CETSP-14-02) to 7.64% (CETSP-18-08). On 18 out of 40 scenarios this gap is greater than 5% and in 7 cases it is greater than 6%. These results highlight the relevant contribution given by Convex Hull strategy to the quality of lower bounds computed by ULB algorithm.

The LB values found by Basic and by Basic+EL are the same because the Elastic Force algorithm does not affect the lower bounds. The upper bounds computed by Basic+EL are better than the ones computed by Basic with a percentage gap that ranges from 1.09% (CETSP-18-08) to 2.07% (CETSP-20-02). The contribution given by the Elastic Force algorithm is not so great, if compared to the contribution given by the Convex Hull strategy for the lower bounds. However it is enough to guarantee better upper bounds for ULB than the ones provided by other algorithms proposed in literature.

Table 2 reports the upper and lower bounds computed by the LB3 algorithm of [2] and by our MIP model on the smallest scenarios CETSP-6. The best bounds are shown in bold. Moreover, the column $Time$ reports the CPU Time, in seconds, required by the two algorithms and the last column $Gap(\%)$ represents the gap, in percentage, between the upper and lower bounds, computed with the formula $\frac{(UB-LB)}{UB}$.

On these scenarios, our algorithm always finds better upper and lower bounds with respect to LB3. Moreover, since the gap values are often lower than 2%, the upper bounds of our algorithm are close to the optimal solution. Instead, the gap values of the LB3 algorithm are often twice bigger than those of the MIP model and, in the worst case (6-05), this gap value is almost 7%. The computational time is negligible for both the algorithms since it is always lower than 2 seconds.

In Table 3 the results of Bender Decomposition (BD) and ULB algorithms are reported and two radius size, $r = 0.25$ and $r = 0.5$ are considered. On the

Scenario	LB3			ULB			Gap (%)	
	UB	LB	Time	UB	LB	Time	LB3	MIP
CETSP-6-01	34.2768	33.1604	0.8	**33.8625**	**33.2953**	0.4	3.3	1.7
CETSP-6-02	27.8568	27.0615	0.8	**27.6407**	**27.2558**	0.7	2.9	1.4
CETSP-6-03	25.6439	24.7048	1.6	**25.3920**	**24.9301**	0.3	3.7	1.8
CETSP-6-04	37.0838	36.1009	0.5	**36.7576**	**36.2682**	0.3	2.7	1.3
CETSP-6-05	22.9133	21.3865	1.2	**22.1507**	**21.5609**	0.3	6.7	2.7
CETSP-6-06	28.1387	27.1195	0.8	**27.9028**	**27.3950**	0.3	3.6	1.8
CETSP-6-07	35.0102	33.9480	0.7	**34.7630**	**34.2264**	0.3	3.0	1.5
CETSP-6-08	24.5140	23.4709	1.5	**24.1722**	**23.6503**	0.3	4.3	2.2
CETSP-6-09	29.0243	27.8231	0.8	**28.4734**	**27.8739**	0.3	4.1	2.1
CETSP-6-10	34.9533	33.6742	0.9	**34.7182**	**34.0544**	0.3	3.7	1.9

Table 2: The results of LB3 and ULB algorithms on the smallest scenarios.

scenarios with $r = 0.25$, ULB is more effective and faster than BD. Indeed, on all the scenarios the gaps given by ULB are lower than the gaps of BD and often there is a 50% of difference between them. Moreover, the gaps of ULB range from 4.0% to 7.0% while the gaps of BD range from 7.8% to 13.7%. This shows that the bounds produced by our algorithm are much better than the bounds of BD. Regarding the performance, ULB is almost always faster than BD and on the largest instances with 20 target points the difference in terms of computation time starts to be significant.

Obviously, the greater is the radius r the higher is the complexity of the scenarios and then we expect worst results of the algorithms when $r = 0.5$. The values in the Table 3 confirm our expectation because the gap of both ULB and BD are doubled. This means that again the gap values of ULB are the half of the gap values of BD. It is interesting to observe that, with respect to ULB, the performance of BD looks heavily affected by the radius value. The computational time spent by ULB to solve an instance with $r = 0.5$ is usually two times the time spent for the same scenario but with $r = 0.25$. Instead, for BD the situation is much different with scenarios solved by spending ten or hundred times the computational time required for the same scenario with $r = 0.25$. Actually there are three scenarios (18-08, 20-07 and 20-08) where BD reaches the time limit of 1500 seconds while ULB solves these scenarios in less than 80 seconds. Moreover, there are several scenarios (16-09, 18-02, 18-

Scenario	r=0.25				r=0.5			
	BD		ULB		BD		ULB	
	Gap	Time	Gap	Time	Gap	Time	Gap	Time
CETSP-14-01	9.3	2.5	**4.4**	2.6	21.3	5.3	**9.3**	3.8
CETSP-14-02	9.5	1.5	**4.5**	1.3	19.8	2.2	**10.1**	3.2
CETSP-14-03	9.2	6.4	**4.3**	1.6	18.9	11.7	**8.6**	4.2
CETSP-14-04	8.5	2.6	**4.5**	1.3	18.6	4.5	**8.9**	3.4
CETSP-14-05	9.3	2.1	**4.2**	1.1	16.3	3.3	**8.6**	1.5
CETSP-14-06	9.6	3.7	**4.9**	2.0	21.9	2.5	**10.0**	3.3
CETSP-14-07	7.8	2.4	**4.0**	1.0	18.9	2.8	**8.7**	2.2
CETSP-14-08	9.8	6.3	**4.7**	1.3	20.2	7.4	**9.9**	2.9
CETSP-14-09	11.5	2.9	**5.0**	1.4	22.3	1.9	**10.0**	3.5
CETSP-14-10	10.2	3.4	**4.9**	2.0	20.5	6.0	**10.3**	4.5
CETSP-16-01	9.4	5.0	**4.7**	2.4	21.9	8.3	**9.5**	7.1
CETSP-16-02	11.5	1.8	**5.6**	1.1	22.8	8.7	**11.8**	3.5
CETSP-16-03	10.9	6.7	**5.2**	1.5	22.1	9.7	**10.6**	5.3
CETSP-16-04	9.7	3.0	**5.0**	1.3	20.0	13.8	**10.9**	4.8
CETSP-16-05	11.1	3.2	**5.7**	2.3	25.0	8.4	**12.0**	5.9
CETSP-16-06	8.3	4.0	**4.3**	3.2	19.9	14.2	**9.3**	3.6
CETSP-16-07	9.9	7.5	**4.7**	3.1	21.6	15.6	**10.3**	14.5
CETSP-16-08	9.9	2.0	**5.2**	1.1	19.9	7.2	**10.7**	2.5
CETSP-16-09	12.0	11.6	**5.8**	3.0	25.9	520.1	**12.0**	10.6
CETSP-16-10	11.7	5.2	**5.6**	2.2	24.8	15.8	**11.5**	6.8
CETSP-18-01	10.3	8.1	**5.1**	4.0	24.0	35.6	**10.7**	8.4
CETSP-18-02	12.8	12.8	**5.7**	4.6	24.3	202.0	**12.0**	8.9
CETSP-18-03	10.7	9.1	**5.6**	4.7	24.5	102.9	**12.2**	13.9
CETSP-18-04	11.8	2.7	**5.2**	2.4	22.7	11.0	**11.0**	4.7
CETSP-18-05	11.7	4.7	**5.8**	3.9	22.5	28.0	**11.7**	5.7
CETSP-18-06	12.5	13.4	**5.8**	5.4	28.0	173.9	**12.6**	11.9
CETSP-18-07	13.5	4.2	**6.1**	3.3	27.7	11.4	**12.2**	7.3
CETSP-18-08	12.9	22.0	**6.3**	4.5	33.4	1501.0	**13.2**	12.5
CETSP-18-09	13.0	5.7	**6.3**	2.2	27.3	38.4	**13.3**	9.0
CETSP-18-10	10.4	4.7	**5.2**	3.9	22.8	36.2	**11.4**	9.8
CETSP-20-01	11.9	11.7	**5.6**	4.0	23.8	556.4	**12.1**	12.5
CETSP-20-02	13.2	11.8	**6.1**	5.1	28.7	135.4	**13.2**	10.0
CETSP-20-03	10.2	8.3	**5.3**	1.9	24.6	41.5	**11.7**	13.6
CETSP-20-04	12.6	9.2	**6.5**	5.2	28.1	84.6	**14.1**	11.0
CETSP-20-05	12.3	8.2	**6.2**	4.4	25.7	716.9	**13.3**	8.4
CETSP-20-06	11.7	15.0	**5.7**	4.0	24.4	177.3	**12.2**	11.1
CETSP-20-07	13.7	15.8	**7.0**	6.8	31.9	1501.3	**14.9**	23.5
CETSP-20-08	13.4	39.4	**6.6**	7.5	30.8	1501.0	**14.0**	78.5
CETSP-20-09	11.5	17.4	**5.8**	3.1	27.3	57.0	**12.7**	8.3
CETSP-20-10	11.3	10.1	**5.8**	5.7	26.6	21.4	**12.8**	10.6

Table 3: Test results of the Bender Decomposition and ULB algorithms on the scenarios up to 20 target points.

Scenario	r=0.5				r=1			
	IA		ULB		IA		ULB	
	LB	Time	LB	Time	LB	Time	LB	Time
CETSP-12-01	**32.308**	17.6	31.185	2.7	23.913	1000.0	**25.849**	26.7
CETSP-12-02	**43.535**	4.3	42.304	1.0	**37.560**	1000.0	35.251	3.7
CETSP-12-03	**31.225**	6.8	30.419	1.0	**25.587**	1000.0	24.08	3.4
CETSP-12-04	**31.792**	16.5	30.607	3.4	23.795	1000.0	**24.920**	11.4
CETSP-12-05	**36.059**	10.4	35.003	2.1	**27.768**	429.5	26.256	8.2
CETSP-12-06	**34.346**	2.9	33.157	2.2	**29.670**	150.2	28.536	7.6
CETSP-12-07	**32.689**	3.5	31.89	2.1	25.902	1000.0	**26.199**	4.7
CETSP-12-08	**36.986**	3.4	35.739	0.9	**31.105**	812.6	28.771	3.6
CETSP-12-09	**30.938**	11.1	29.447	2.3	**23.908**	1000.0	22.412	5.3
CETSP-12-10	**38.745**	4.4	37.537	1.6	**31.166**	439.2	30.155	3.0

Table 4: Lower bounds computed by ULB and by IA on the scenarios with 12 target points.

06, 18-08, 20-01, 20-02, 20-05, 20-06, 20-07, 20-08) where ULB is an order of magnitude faster than BD.

The last set of scenarios is composed by 12 target points and two radii $r = 0.5$ and $r = 1$. The first comparison is carried out on the lower bounds computed by ULB and IA and the results are reported in Table 4. For $r = 0.5$ all the lower bounds computed by IA are better than the lower bounds of our algorithm. This is not surprising because, as explained in the previous section, our approach is developed to compute tighter upper bounds while the lower bounds are just computed by subtracting the discretization error of each neighborhood from the upper bound. Regarding the performance ULB is always faster than IA but the differences in computational times are not particularly relevant. The situation changes significantly when $r = 1$. We observe that IA is less effective on these scenarios and, indeed, three times (12-01, 12-04, 12-07) ULB finds the best lower bound. What is really impressive is the decay of the performance of IA that reaches the time limit of 1000 seconds on six scenarios and requires from 150 to 800 seconds to solve the remaining four scenarios. Instead, all these scenarios are solved by ULB in less than 4 seconds. These results highlight that the radius size heavily affects the performance of IA algorithm, as already shown for the BD algorithm in Table 3, while for ULB is fairly less relevant since all scenarios with $r = 1$ are solved in less than 30 seconds.

Finally, in Table 5 the upper bound values computed by IA and ULB are

Scenario	r=0.5				r=1			
	IA		ULB		IA		ULB	
	UB	Time	UB	Time	UB	Time	UB	Time
CETSP-12-01	34.113	224.9	**34.113**	2.7	**31.398**	1000.0	31.300	26.7
CETSP-12-02	45.235	14.8	**45.216**	1.0	41.051	1000.0	**41.044**	3.7
CETSP-12-03	33.204	37.6	**33.169**	1.0	29.458	1000.0	**29.444**	3.4
CETSP-12-04	33.253	97.0	**33.260**	3.4	**29.880**	1000.0	29.861	11.4
CETSP-12-05	38.077	205.7	**38.046**	2.1	32.623	1000.0	**32.584**	8.2
CETSP-12-06	36.163	6.6	**36.154**	2.2	34.192	1000.0	**34.125**	7.6
CETSP-12-07	35.092	34.4	**35.055**	2.1	**31.988**	1000.0	32.138	4.7
CETSP-12-08	38.410	9.8	**38.393**	0.9	34.191	1000.0	**34.151**	3.6
CETSP-12-09	32.572	81.8	**32.558**	2.3	28.688	1000.0	**28.672**	5.3
CETSP-12-10	40.716	32.1	**40.702**	1.6	36.589	1000.0	**36.543**	3.0

Table 5: Upper bounds computed by ULB and by IA on the scenarios with 12 target points.

reported. For $r = 0.5$ the upper bounds of ULB are always better than the upper those of IA and they are computed in less time. In particular often ULB is an order of magnitude faster then IA. By increasing the radius to 1, ULB is less effective and IA finds the best upper bound three times. However, on these scenarios IA always reaches the time limit of 1000 seconds while ULB remains under 30 seconds. Obviously, we could obtain better bounds by increasing the value k of the discretization points but we prefer to have a very fast algorithm producing reasonably good bounds, in view of possible embedding of ULB into exact approaches, where it is crucial to quickly generate good solutions.

In order to evaluate how the running time of ULB is affected by problem size, we generated a set of larger instances, with 25 and 30 targets, according to the procedure reported in [2]. More in details, the target locations and the depot are chosen randomly on a rectangle with length equal to 16 and width equal to 10. The neighborhoods of the targets are discs of identical radius r=0.5. The results of ULB on these new instances are reported in Table 6.

In this Table the first column identifies the scenario while the second and third ones report the upper bounds computed by TSP and TSP+EL algorithms, respectively. TSP algorithm computes the shortest tour that starts from the depot and visits all target points. Instead, TSP+EL applies the elastic force algorithm on the solution found by TSP to derive a better solution. The next

Scenario	TSP UB	TSP+EL UB	ULB UB	ULB LB	ULB Time	Gap UB
CETSP-25-01	55.389	44.621	42.854	35.923	95.630	3.96%
CETSP-25-02	59.395	49.759	47.321	40.687	16.644	4.90%
CETSP-25-03	56.628	48.473	45.281	38.787	102.008	6.59%
CETSP-25-04	58.218	48.466	47.412	40.261	21.188	2.17%
CETSP-25-05	58.107	48.974	46.364	39.449	55.106	5.33%
CETSP-25-06	56.478	48.252	45.564	38.550	93.906	5.57%
CETSP-25-07	59.830	49.993	49.032	42.140	28.697	1.92%
CETSP-25-08	57.853	50.022	48.264	41.607	57.222	3.51%
CETSP-25-09	54.967	46.121	43.366	36.684	132.461	5.97%
CETSP-25-10	56.194	47.107	44.823	37.585	49.672	4.85%
CETSP-30-01	62.132	51.877	48.844	40.108	82.275	5.85%
CETSP-30-02	58.520	46.599	44.936	36.487	78.372	3.57%
CETSP-30-03	63.318	53.478	51.463	43.300	334.090	3.77%
CETSP-30-04	60.218	49.173	48.489	39.124	866.008	1.39%
CETSP-30-05	57.776	48.836	46.572	38.117	129.343	4.64%
CETSP-30-06	62.263	52.599	50.333	41.886	56.073	4.31%
CETSP-30-07	63.613	52.042	49.914	41.640	537.379	4.09%
CETSP-30-08	63.478	54.583	50.723	42.489	76.075	7.07%
CETSP-30-09	60.151	49.952	48.492	39.744	331.810	2.92%
CETSP-30-10	59.575	50.387	47.686	39.251	232.453	5.36%

Table 6: Computational results of ULB and of TSP+EL algorithms on the the larger instances with r=0.5.

three columns of the table report the upper and lower bounds and the computational time of ULB, respectively. Finally, the last column shows the percentage gap between the upper bounds computed by TSP+EL and ULB.

The comparison between TSP and TSP+EL allows us to further investigate the effectiveness of the elastic force algorithm. It is evident from the results that the upper bounds found by TSP+EL are much better than the upper bounds of TSP with a percentage gap that ranges from 13% to 20%. These values highlight the effectiveness of the elastic force algorithm that is able to produce good upper bounds starting from the tsp solution whose value is usually far from the optimal solution value.

Despite the remarkable improvement obtained by elastic force algorithm, it is not sufficient to reach the upper bounds computed by ULB algorithm as

shown by results of Gap column. From this last column it is apparent that the ULB solutions are always better than the TSP+EL solutions and the percentage gap ranges from 1.39% (CETSP-30-03) to 7.07% (CETSP-30-07).

Regarding the performance, obviously TSP+EL is very fast and its computational time is negligible because lower than 1 second. For this reason, we did not report this value into the table. Instead, for the ULB algorithm we observe a computational time increase, in particular on the scenarios with 30 targets where the GTSP has to be solved on graphs with 240 discretization points. All the scenarios with 25 target points, except CETSP-25-09, are solved in less than two minutes. Instead, on the scenarios with 30 target points, the computational time ranges from 1 to 15 minutes. These results show that, as expected, the greater the size of the problem, the greater is the computational time required by ULB due to the resolution of the GTSP problem. Anyway, it is always possible to find an appropriate trade-off between the effectiveness and the performance of ULB algorithm by increasing or by reducing the number of discretization points to be used for each neighborhood.

8. Conclusion

In this work we studied the Close Enough Traveling Salesman Problem and developed an approach to compute upper and lower bounds for the optimal solution. The main contribution of our work is the introduction of a new discretization scheme that, by reducing the discretization error, improves the quality of the upper and lower bounds found. Moreover, we introduced an effective graph reduction algorithm. The numerical experiments demonstrate that our approach significantly outperform previous approaches to CETSP, in terms of both computational time and quality of the bounds.

References

[1] E. M. Arkin, R. Hassin, Approximation algorithms for the geometric covering salesman problem, Discrete Applied Mathematics 55 (1995) 197–218.

[2] B. Behdani, J. Smith, An integer-programming-based approach to the close-enough traveling salesman problem, INFORMS Journal on Computing 26 (3) (2014) 415–432.

[3] T. Cormen, C. Leiserson, R. R.L., S. C., Introduction to Algorithms, MIT Press, 2009.

[4] J. Dong, N. Yang, M. Chen, Heuristic approaches for a tsp variant: The automatic meter reading shortest tour problem, Operations Research/ Computer Science Interfaces Series 37 (2007) 145–163.

[5] A. Dumitrescu, J. Mitchell, Approximation algorithms for tsp with neighborhoods in the plane, J. Algorithms 48 (1) (2003) 135–159.

[6] M. Fischetti, J. Salazar-Gonzalez, P. Toth, The generalized traveling salesman and orienteering problems, in: The Traveling Salesman Problem and Its Variations, vol. 12 of Combinatorial Optimization, Springer US, 2007, pp. 609–662.

[7] D. Gulczynski, J. Heath, C. Price, Close enough traveling salesman problem: A discussion of several heuristics, in: Perspectives in Operations Research, vol. 36 of Operations Research/Computer Science Interfaces Series, Springer US, 2006, pp. 271–283.

[8] G. Gutin, D. Karapetyan, Generalized traveling salesman problem reduction algorithms, Algorithmic Operations Research 4 (2009) 144–154.

[9] C. S. Mata, J. S. B. Mitchell, Approximation algorithms for geometric tour and network design problems (extended abstract), in: Proceedings of the Eleventh Annual Symposium on Computational Geometry, SCG '95, ACM, New York, NY, USA, 1995, pp. 360–369.

[10] W. Mennell, Heuristics for solving three routing problems: Close-enough traveling salesman problem, close-enough vehi- cle routing problem, sequence-dependent team orienteering problem, Ph.D. thesis, The Robert H. Smith School of Business, University of Maryland, College Park. (2009).

[11] W. Mennell, B. Golden, E. Wasil, A steiner-zone heuristic for solving the close-enough traveling salesman problem, in: 2th INFORMS Computing Society Conference: Operations Research, Computing, and Homeland Defense, 2011.

[12] R. Shuttleworth, B. Golden, S. Smith, E. Wasil, Advances in meter reading: Heuristic solution of the close enough traveling salesman problem over a street network, Operations Research/ Computer Science Interfaces Series 43 (2008) 487–501.

[13] B. Yuan, M. Orlowska, S. Sadiq, On the optimal robot routing problem in wireless sensor networks, IEEE Transactions on Knowledge and Data Engineering 19 (9) (2007) 1252–1261.