Improved upper and lower bounds for the close enough traveling salesman problem

Francesco Carrabs, Carmine Cerrone, Raffaele Cerulli, and Ciriaco D'Ambrosio

[1] Department of Mathematics, University of Salerno
{fcarrabs,raffaele,cdambrosio}@unisa.it
[2] Department of Biosciences and Territory, University of Molise
{carmine.cerrone}@unimol.it

Abstract. This paper studies the close-enough traveling salesman problem, a variant of the Euclidean traveling salesman problem in which the traveler visits a node if it passes through the neighborhood of that node. We introduce an improved version of the adaptive internal discretization scheme, recently proposed in the literature, and a heuristic that combines this scheme with to a second-order cone programming algorithm. Our heuristic is able to compute tight bounds for the problem. The computational results, carried out on benchmark instances, confirm the improvements of the bounds computed with respect to the other algorithms proposed in the literature.

Keywords: Close-Enough, Traveling Salesman Problem, Discretization scheme, Second-Order Cone Programming

1 Introduction

This paper concerns a variant of the traveling salesman problem (TSP) called close-enough traveling salesman problem (CETSP). The TSP is one of the most studied optimization problems. Given a set of target points, in an Euclidean space, the TSP consists of finding a minimum length tour that starts and ends at a depot while visiting each target point exactly once. In CETSP to each target point v is associated a neighborhood, that is a compact region of the space containing v. The CETSP consists of finding the shortest tour that starts and ends at the depot and intersects each neighborhood once. In this work we assume that the neighborhoods shape is a disc.

The CETSP has several practical applications in the context of Unmanned Aerial Vehicles that are vehicles without crew used for military and civil missions like aerial forest fire detection, supply delivering (food, munition, etc.) to targets, geographic region monitoring and military surveillance. Moreover, even the robot monitoring wireless sensor networks can be modeled as a CETSP [15]. Finally, the CETSP arises in the context of the reading process of water (electricity or gas) consumption. Indeed, by using radio frequency identification readers, it is possible to catch the information about the consumption by using a drone

flying within the range of each reader. This methodology is much faster than the classical door-by-door reading process and requires the resolution of CETSP.

The CETSP was introduced by Gulczynski et al. [12] that proposed several heuristics to face the problem. Afterwards, other heuristics were introduced in [10, 13, 14] while an effective evolutionary approach was implemented in [15].

Very recently, a branch-and-bound algorithm was proposed in [9] in which the subproblem, solved in each node of the search tree, consists of a Second Order Cone Programming. The authors carried out computational tests on several instances proving that their branch and bound often finds the optimal solution. To the best of our knowledge, the idea to use the second order cone programming for the CETSP was introduced the first time in [14].

In this paper we propose a more effective version of the internal discretization scheme introduced in [5]. Moreover, we propose a new heutistic obtained by combining this new scheme with the second-order cone programming algorithm [14] to obtain a heuristic able to compute tight bounds for the CETSP problem. The computational results reveal that our approach produces better bounds with respect to the ones proposed in [2, 5].

The remainder of the paper is organized as follows. Section 2 introduces the definitions and the notations that are throughout the paper. Section 3 contains a brief overview of the algorithm based on the internal discretization scheme [5] while Section 4 reports a mixed-integer programming model for CETSP. The second-order cone programming algorithm [14] is described in Section 5. Finally, computational results are presented in Section 6 followed by conclusions in Section 7.

2 Definitions and Notation

Given a two-dimensional plane, let N be a set of *target points* placed in the plane, with $|N| = n$, and let $v_0 \notin N$ be the depot point. A circumference C_v, with center v and radius r_v, is associated to each target point $v \in N$ (Figure 1(a)). The set of points within and on C_v compose the *neighborhood* $N(v)$ of v. W.l.o.g., we suppose that $v_0 \notin N(v), \forall v \in N$. Given the neighborhood $N(v)$ depicted in Figure 1(a), let d_i and d_j be two points on 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.

The CETSP consists in finding a shortest tour T^* that starts and ends into the depot v_0 and intersects the neighborhoods of all target points. Given a tour T, its *turn points* are the points where a direction change occurs. For instance, in Figure 1(b) it is shown a feasible tour for the CETSP with the turn points v_0, p_1, p_2 and p_3. Note that any tour can be uniquely identified through the sequence of its turn points. 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 and it is denoted by $w(\overline{p_i, p_j})$. 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.

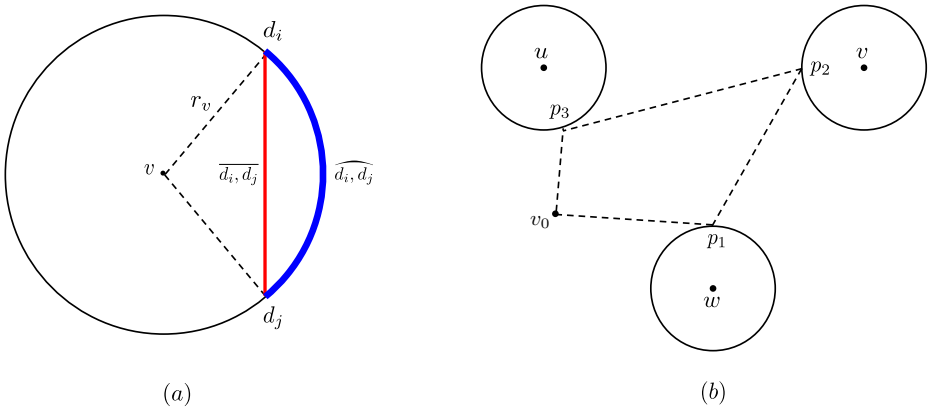

Fig. 1. (a) Circumference C_v of the target point v. The points in and on C_v compose the neighborhood $N(v)$. (b) A feasible tour, for the CETSP, composed by turn points v_0, p_1, p_2 and p_3.

3 The Adaptive Internal Discretization Algorithm

In this section we briefly describe the *internal point discretization* scheme (IP) and the steps of the *internal discretization algorithm*, based on IP, that finds upper bounds for CETSP. See [5] for a detailed explanation of both the discretization scheme and the algorithm.

It is easy to see that the number of feasible tours for CETSP is infinite because each neighborhood $N(v)$, with $v \in N$, contains an infinite number of turn points usable to create the tours. However, any single tour can be identified by using a finite set of turn points. For this reason, we discretize the neighborhoods by using a finite number of discretization points and we use only these points to build the tours. Obviously, this choice implies that our algorithm will consider only a subset of infinite feasible tours of the problem. In more details, each neighborhood $N(v)$ is discretized by using a fixed number k of discretization points denoted by $\hat{N}(v)$. After the discretization, we build the *discretized graph* $G = (V, E)$, where $V = \bigcup_{v \in N} \hat{N}(v)$ and $E = \{(x, y) : x \in N(u), y \in N(v), u \neq v\}$.

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. It is easy to see that the weight of any tour $\hat{T}$ of G, that starts and ends at the depot and that visits exactly one discretization point in each neighborhood, is an upper bound to $w(T^*)$. However, in order to find tighter upper bound of $w(T^*)$ we solve the Generalized Traveling Salesman Problem (GTSP) [11] on G obtaining the shortest tour $\hat{T}^*$. The tightness of $w(\hat{T}^*)$ with respect to $w(T^*)$ heavily depends i) on the number of points k used to carry out the discretization and ii) on their placement in each neighborhood. Obviously, as the number of discretization points increases as the quality of the upper bound improves. However, by

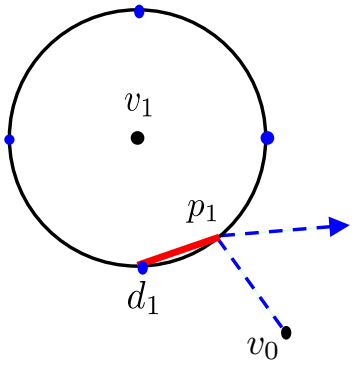

Fig. 2. The discretization error carried out on $\hat{N}(v_1)$. p_1 is the turn point of T^* while d_1 is the discretization point used by $\hat{T}^*$. The length of $\overline{p_1, d_1}$, multiplied by 2, corresponds to $\xi(v_1)$.

increasing the discretization points used we increase the size of the graph G too and then the computational time required to solve the GTSP on G. For this reason, the number of discretization points, to use for each neighborhood, has to be wisely defined in order to obtain an appropriate trade-off between the quality of the upper bound and the time spent to compute it.

Fixed the number k of discretization point to use for each neighborhood the next question is: where are the best positions where to place these discretization points in order to obtain an upper bound as tight as possible? There are various *discretization scheme*, proposed in literature, that state the positions of the discretization points in each neighborhood. In this work we use IP scheme introduced in [5]. However, before describing this scheme, let us see why the positions of discretization points can significantly change the quality of the upper bound computed.

Since the construction of $\hat{T}^*$ is carried out by using only the discretization points then a *discretization error* $\xi(v_i)$ occurs, on each neighborhood $\hat{N}(v_i)$, with respect to the turn point p_i of T^* in $N(v_i)$. If d_i is the discretization point of $\hat{N}(v_i)$ closest to the turn point p_i, then $\xi(v_i)$ is equal to two times the length of $\overline{p_i, d_i}$. For instance, in Figure 2 the neighborhood $N(v_1)$ is discretized by using four points that are placed on C_{v_1}. Since the tour T^* intersects $N(v_1)$ in the turn point p_1 then $\xi(v_i)$ is equal to two times the length of $\overline{p_1, d_1}$, one time to come from p_1 to d_1 and another one to come back. Let Q be a walk composed by edges of T^* and by discretization errors of all the neighborhoods. This means that $w(Q) = w(T^*) + \sum_{v \in N} \xi(v)$. Moreover, since Q starts and ends at the depot v_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 $\sum_{v \in N} \xi(v)$. Of course, the best situation occurs when $\xi(v_i) = 0$ that is when p_1 coincides with d_1 in

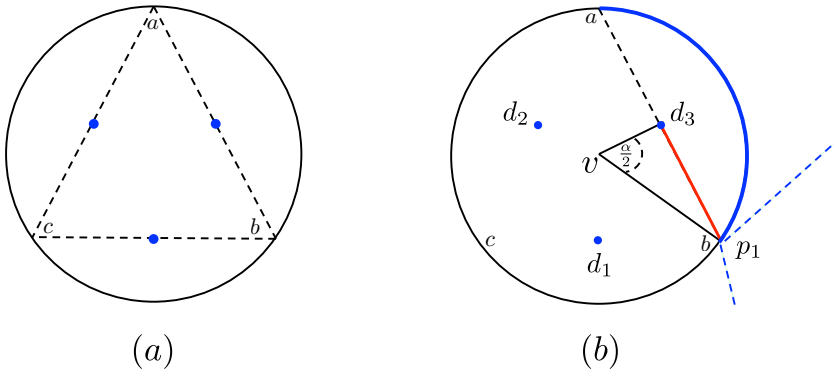

Fig. 3. (a) The internal discretization scheme for $k = 3$. (b) Computation of the maximum error $\xi(v)$ for IP scheme.

Figure 2 but, in general, $\xi(v_i) > 0$ and it will affect the quality of the upper bounds found.

3.1 Internal Point Discretization Scheme

The internal point discretization (IP) scheme was introduced in [5] with the aim to minimize the discretization error carried out during the construction of $\hat{T}^*$ whatever are the turn points of T^*. Given k discretization points, the IP scheme divides C_v in k equal circular arcs and, for each arc $\overset{\frown}{a,b}$, places a discretization point in the middle of the chord $\overline{a,b}$. In Figure 3(a) the IP scheme is shown for $k = 3$.

In order to compute the discretization error $\xi(v)$, produced by IP scheme, let us consider the example in 3(b). If the turn point p_1 of T^* intersects $N(v)$ on the circular arc $\overset{\frown}{a,b}$ then the maximum distance between p_1 and the discretization point d_3 occurs when p_1 coincides with the point a or b. Therefore, the discretization error $\xi(v)$ is equal to $2(\frac{1}{2}w(\overline{a,b})) = w(\overline{a,b})$. The same reasoning holds for the circular arc $\overset{\frown}{b,c}$ with the discretization point d_1 and for the circular arc $\overset{\frown}{c,a}$ with the discretization point d_2.

Now, from trigonometry, we know that $w(\overline{a,b}) = 2r_v \sin(\frac{\alpha}{2})$, where α is the central angle associated to the circular arc $\overset{\frown}{a,b}$. Note that, by definition, $\alpha = \frac{2\pi}{k}$. As consequence, fixed the number k of discretization points, the maximum error associated to the IP scheme in $\hat{N}(v)$ is $\xi(v) = 2r_v \sin(\frac{\pi}{k})$.

3.2 Convex Hull Strategy

In [2] the authors proved that the turn points of T^* are always on the circular arcs belonging to the convex hull generated by target points. This means that all the points of the circular arcs outside $conv(N \cup \{v_0\})$ can be discarded because they cannot belong to the optimal solution. In Figure 4 the convex hull, generated

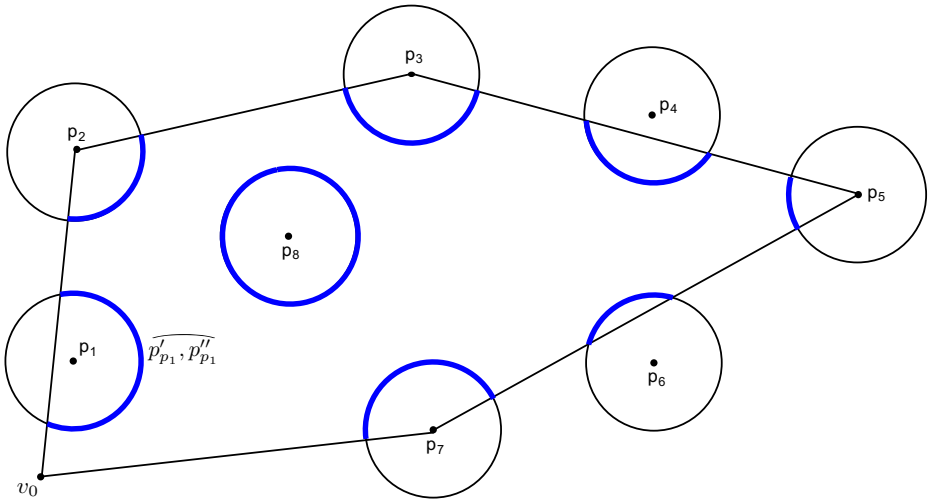

Fig. 4. In blue the circular arcs discretized after the construction of the convex hull.

by depot and target points $p_1, \ldots, p_8$ is depicted. The circular arcs within the convex hull are highlighted in blue and, from now on, are referred to as *feasible circular arcs* (*fca*). 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 $\widehat{v'_i, v''_i}$.

Since the turn points of T^* can be only on the *fca*, rather than wasting discretization points on whole circumference C_v, in [5] the author discretized only the *fca*. Moreover, rather than using the same number k of points for each *fca*, they assigned the number of discretization points to be used for each *fca* according to its length. For instance, the length of $\widehat{p'_{p_1}, p''_{p_1}}$, in Figure 4, is much longer than the length of $\widehat{p'_{p_5}, p''_{p_5}}$ and then the number of discretization points assigned to the former *fca* should be greater than the number of discretization points assigned to the latter *fca* in order to minimize the discretization errors.

The number of discretization points, assigned to each neighborhood, is defined as follows. Given a neighborhood $N(v_i)$, 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 $\left\lfloor \frac{\alpha_i}{\hat{\alpha}} \right\rfloor$ points. According to this idea, the discretization error $\xi(v_i)$, carried out on $\hat{N}(v_i)$, is expressed as: $\xi(v_i) = 2r_{v_i} \sin(\frac{\alpha_i}{2k_i}) = 2r_{v_i} \sin(\frac{\hat{\alpha}}{2})$ where $k_i = \frac{\alpha_i}{\hat{\alpha}}$.

Here we introduce a new modification to the previous scheme that can significantly improve the lower bounds computed. The idea behind this modification is the following. Since k_i is an integer value and $k_i = \frac{\alpha_i}{\hat{\alpha}}$ then a truncation operations is often carried out when we compute k_i. Due to this truncation, the total number of discretization points used could be lower than $k|N|$. When this

event occurs, we have discretization points that can be used to further reduce the discretization errors. Let ℓ be the number of remaining discretization points available. Then we apply a greedy algorithm, named *residual points assignment* (RPA) to assign even these points. The pseudocode of RPA is reported in Algorithm 1.

Algorithm 1 Residual Points Assignment

1: Input: The value ℓ.
2: **while** $\ell > 0$ **do**
3: Select vertex v_i such that $\xi(v_i) \geq \xi(v_j) \quad \forall j \in N$
4: $k_i \leftarrow k_i + 1$
5: $\ell \leftarrow \ell - 1$
6: **end while**

For each residual point available, the algorithm finds the vertex v_i with maximum discretization error (line 3) and try to reduce this last error by increasing by one the number of discretization point used for $\overset{\frown}{p'_{v_i}, p''_{v_i}}$ (line 4). The process is repeated until all the residual points are assigned.

4 Mathematical Formulation

In this section we report the mathematical formulation of the GTSP problem that is used to compute $\hat{T}^*$. Since the resolution of GTSP is expensive, we reduce the complexity of the problem by reducing the size of the graph $G(V, E)$. To this end, we apply on G the graph reduction algorithm described in [5]. This algorithm looks for useless edges in G, that is edges that cannot belong to $\hat{T}^*$, and it removes them. Often the number of edges in G is reduced by 50%, thanks to this algorithm, making the individuation of $\hat{T}^*$ less expensive.

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:

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

The objective function (1) minimizes the cost of the tour. Constraints (2) guarantee that a discretization point of each neighborhood is visited while Constraints (3) ensure that at most one edge connecting two neighborhoods is selected. Constraints (4) 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 (5) are the subtour elimination constraints adapted to the Generalized TSP [11].

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, $LB = w(\hat{T}^*) - \sum_{v\in V} \xi(v)$.

Finally, once the solution $\hat{T}^*$ has been found, we carry out an additional step to improve the upper bound, this additional step is introduced in the next section.

5 Second-Order Cone Programming Algorithm

The solution $\hat{T}^*$ computed by MIP model in the previous section, can be further improved by modifying the position of the discretization points used. More in details, when the visiting sequence of target points is fixed a priori, the CETSP corresponds to the Touring Steiner Zones Problem (TSZP). This problem can be formulated as a second-order cone programming (SOCP) [13] and solved in polynomial time [1]. In the following we briefly describe the formulation proposed in [13] and that is implemented in our algorithm to further improve the upper bounds computed.

$$(\mathbf{SOCP}) \qquad \min \sum_{i=0}^{|N|} z_i \tag{6}$$

$$w_i = x_i - x_{i+1} \qquad \forall\, i \in \{0, \dots, |N|\} \tag{7}$$

$$u_i = y_i - y_{i+1} \qquad \forall\, i \in \{0, \dots, |N|\} \tag{8}$$

$$s_i = \bar{x}_i - x_i \qquad \forall\, i \in \{0, \dots, |N|\} \tag{9}$$

$$t_i = \bar{y}_i - y_i \qquad \forall\, i \in \{0, \dots, |N|\} \tag{10}$$

$$z_i^2 \geq w_i^2 + u_i^2 \qquad \forall\, i \in \{0, \dots, |N|\} \tag{11}$$

$$s_i^2 + t_i^2 \leq r_i^2 \qquad \forall\, i \in \{0, \dots, |N|\} \tag{12}$$

$$z_i \geq 0 \qquad \forall\, i \in \{0, \dots, |N|\} \tag{13}$$

$$w_i,\ u_i,\ s_i,\ t_i,\ x_i,\ y_i,\ free \qquad \forall\, i \in \{0, \dots, |N|\} \tag{14}$$

The tour $\hat{T}^*$, computed by MIP, defines a visiting sequence of the target points. W.l.o.g. let us suppose that such a sequence is $(v_1, v_2, \dots, v_{|N|})$. We replicate the first target point v_1 in the last position of the sequence obtaining the new sequence $(v_1, v_2, \dots,\ v_{|N|}, v_1)$. To each target point v_i in the sequence are associated the variables x_i and y_i representing the coordinates of the discretization point that covers v_i in the tour. For each target point v_i, we want to find the best position where to place the discretization point in order to minimize the tour length. The z_i variables represents the Euclidean distance between the discretization points associated to v_i and v_{i+1}. The objective function (6) minimizes the sum of Euclidean distances among the discretization points of the tour. Finally, the variables w_i and u_i are used in constraints (11), to compute the Euclidean distance between the vertex v_i and v_{i+1} while the variables s_i and t_i are used in constraints (12), to ensure that the position of discretization point (x_i, y_i) is inside the circumference C_{v_i}.

6 Computational Results

This section presents the computational results of our algorithm, named IULB on the largest benchmark instances proposed in [2] and [5]. IULB 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. In order to verify both the effectiveness and the performance of IULB we will compare it with the Bender Decomposition (BD) proposed in [2] and the ULB algorithm proposed in [5].

In Table 1 the results of BD, ULB and IULB algorithms, on the instances from 14 to 20 target points and with a radius equal to 0.25 and 0.5, are reported. To each algorithm are associated two columns: Gap and Time. The Gap value represents the gap in percentage between the upper (UB) and lower (LB) bound values and it is computed with the formula: $100 \times \frac{(UB-LB)}{UB}$. The Time value is

Instance	r=0.25						r=0.5					
	BD		ULB		IULB		BD		ULB		IULB	
	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time
CETSP-14-01	9.3	2.5	4.4	2.6	4.3	2.1	21.3	5.3	9.3	3.8	9.0	3.8
CETSP-14-02	9.5	1.5	4.5	1.3	4.4	1.2	19.8	2.2	10.1	3.2	9.7	2.4
CETSP-14-03	9.2	6.4	4.3	1.6	4.2	1.2	18.9	11.7	8.6	4.2	8.2	4.2
CETSP-14-04	8.5	2.6	4.5	1.3	4.3	1.0	18.6	4.5	8.9	3.4	8.7	2.5
CETSP-14-05	9.3	2.1	4.2	1.1	4.0	0.7	16.3	3.3	8.6	1.5	8.4	1.7
CETSP-14-06	9.6	3.7	4.9	2.0	4.4	1.5	21.9	2.5	10.0	3.3	9.1	2.6
CETSP-14-07	7.8	2.4	4.0	1.0	3.9	0.8	18.9	2.8	8.7	2.2	8.5	1.8
CETSP-14-08	9.8	6.3	4.7	1.3	4.5	1.1	20.2	7.4	9.9	2.9	9.6	3.0
CETSP-14-09	11.5	2.9	5.0	1.4	4.9	1.2	22.3	1.9	10.0	3.5	9.7	2.8
CETSP-14-10	10.2	3.4	4.9	2.0	4.8	1.1	20.5	6.0	10.3	4.5	10.0	3.3
CETSP-16-01	9.4	5.0	4.7	2.4	4.5	2.1	21.9	8.3	9.5	7.1	9.2	6.2
CETSP-16-02	11.5	1.8	5.6	1.1	5.5	0.9	22.8	8.7	11.8	3.5	11.4	2.5
CETSP-16-03	10.9	6.7	5.2	1.5	5.1	1.3	22.1	9.7	10.6	5.3	10.3	3.9
CETSP-16-04	9.7	3.0	5.0	1.3	4.9	1.1	20.0	13.8	10.9	4.8	10.7	5.2
CETSP-16-05	11.1	3.2	5.7	2.3	5.3	1.9	25.0	8.4	12.0	5.9	11.3	4.9
CETSP-16-06	8.3	4.0	4.3	3.2	4.2	1.9	19.9	14.2	9.3	3.6	8.9	3.4
CETSP-16-07	9.9	7.5	4.7	3.1	4.6	6.1	21.6	15.6	10.3	14.5	10.1	3.8
CETSP-16-08	9.9	2.0	5.2	1.1	4.9	1.0	19.9	7.2	10.7	2.5	10.3	1.3
CETSP-16-09	12.0	11.6	5.8	3.0	5.6	2.3	25.9	520.1	12.0	10.6	11.7	6.6
CETSP-16-10	11.7	5.2	5.6	2.2	5.4	3.5	24.8	15.8	11.5	6.8	11.3	6.9
CETSP-18-01	10.3	8.1	5.1	4.0	4.9	2.8	24.0	35.6	10.7	8.4	10.4	7.6
CETSP-18-02	12.8	12.8	5.7	4.6	5.5	2.3	24.3	202.0	12.0	8.9	11.6	7.8
CETSP-18-03	10.7	9.1	5.6	4.7	5.4	4.3	24.5	102.9	12.2	13.9	11.8	6.6
CETSP-18-04	11.8	2.7	5.2	2.4	5.0	2.0	22.7	11.0	11.0	4.7	10.7	5.1
CETSP-18-05	11.7	4.7	5.8	3.9	5.6	2.2	22.5	28.0	11.7	5.7	11.5	4.2
CETSP-18-06	12.5	13.4	5.8	5.4	5.6	4.4	28.0	173.9	12.6	11.9	12.0	15.2
CETSP-18-07	13.5	4.2	6.1	3.3	5.9	2.7	27.7	11.4	12.2	7.3	11.9	3.5
CETSP-18-08	12.9	22.0	6.3	4.5	6.1	4.1	33.4	1501.0	13.2	12.5	12.8	7.8
CETSP-18-09	13.0	5.7	6.3	2.2	6.1	1.7	27.3	38.4	13.3	9.0	12.9	6.8
CETSP-18-10	10.4	4.7	5.2	3.9	5.1	2.3	22.8	36.2	11.4	9.8	11.1	6.1
CETSP-20-01	11.9	11.7	5.6	4.0	5.4	3.9	23.8	556.4	12.1	12.5	11.7	12.5
CETSP-20-02	13.2	11.8	6.1	5.1	5.9	6.4	28.7	135.4	13.2	10.0	12.7	9.7
CETSP-20-03	10.2	8.3	5.3	1.9	5.1	4.3	24.6	41.5	11.7	13.6	11.3	7.8
CETSP-20-04	12.6	9.2	6.5	5.2	6.1	4.2	28.1	84.6	14.1	11.0	13.6	7.2
CETSP-20-05	12.3	8.2	6.2	4.4	5.9	2.4	25.7	716.9	13.3	8.4	12.7	8.1
CETSP-20-06	11.7	15.0	5.7	4.0	5.5	13.2	24.4	177.3	12.2	11.1	12.0	15.8
CETSP-20-07	13.7	15.8	7.0	6.8	6.8	3.7	31.9	1501.3	14.9	23.5	14.4	15.2
CETSP-20-08	13.4	39.4	6.6	7.5	6.4	8.0	30.8	1501.0	14.0	78.5	13.6	48.8
CETSP-20-09	11.5	17.4	5.8	3.1	5.7	5.0	27.3	57.0	12.7	8.3	12.4	9.5
CETSP-20-10	11.3	10.1	5.8	5.7	5.6	1.9	26.6	21.4	12.8	10.6	12.4	10.7
AVG	**11.0**	**8.0**	**5.4**	**3.1**	**5.2**	**2.9**	**23.8**	**190.1**	**11.4**	**9.4**	**11.0**	**7.2**

Table 1. Test results of the Bender Decomposition, ULB and IULB algorithms on the instances up to 20 target points.

the CPU Time, in seconds, spent by algorithms to compute the bounds. Finally, the average Gap and Time values are reported in the last line of the table.

On the scenarios with $r = 0.25$, IULB is much more effective and faster than BD. Indeed, the Gap values of IULB are always better than the ones of BD and the improvements ranges from 49.4 % (CETSP-16-06) to 57.48% (CETSP-

18-04). With so high improvements, it is easy to state that, at least on these instances, IULB is much more effective than BD. Regarding the performance, IULB is always faster than BD and it results at least 50% faster than BD in 29 out of 40 instances.

More interesting is the comparison between IULB and ULB because allow us to evaluate the impact of ideas proposed in this paper on the effectiveness and the performance of original algorithm ULB. Obviously, due to the kind of ideas used in IULB, its Gap values should be always better than or equal to the Gap values of ULB. For this reason, thanks to the computational tests, we are interested to quantify the quality improvement of the solution found by IULB with respect to ULB and to evaluate the performance of new algorithm. The results show that the Gap values of IULB are always better than the Gap values of ULB. These improvements range from 1.15% (CETSP-14-09) to 9.36% (CETSP-14-06). It is interesting to observe that, despite the use of a greedy algorithm to introduce more discretization points and the application of an exact approach instead of the elastic force heuristic, IULB results slower than ULB only in 4 out of 40 instances (CETSP-16-07, CETSP-20-02, CETSP-20-06, CETSP-20-08) while it is from 14% to 84% faster than ULB in the remaining 36 instances.

By increasing the radius to 0.5 the instances become harder to solve as proven by the increment of both the Gap values and Time of all the algorithms. Again, the Gap values of IULB are always better than the Gap values of BD with an improvement that reaches 61.82% (CETSP-18-08). Hence, the increment of radius amplifies the gap effectiveness between these two algorithms. Regarding the performance, only in three cases BD results faster than IULB but the gap time in these cases is lower than a second and then negligible. On the contrary, on the remaining 37 instances there are cases where IULB is orders of magnitude faster than BD, in particular on the instances with 20 target points. Moreover, there are three scenarios (18-08, 20-07 and 20-08) where BD reaches the time limit of 1500 seconds while IULB solves these scenarios in less than 50 seconds.

Even on the instances with $r = 0.5$ the Gap values of IULB are always better than the Gap values of ULB with the improvements that range from 1.45% to 9.05%. These results are very similar to the ones observed for the case with $r = 0.25$ and then increasing the radius value does not change the effectiveness gap between IULB and ULB. More interesting is to analyze the computational time of IULB on these instances because it appears more expensive. Indeed, as the radius increases as the Time value of IULB increases and there are more instances in which IULB is slower than ULB. However, this gap time never exceeds 5 seconds then it is not so relevant.

Finally, from the average values reported in the last line of the table, it is evident that IULB is the more effective and efficient algorithm. Moreover, these values further certify that the greater is the radius r the higher is the complexity of the instances.

In Table 2 the results of ULB and IULB on the largest instances with 25 and 30 target points and radius $r = 0.5$ are reported. As expected, the Gap values show that as the size of the instances increases as the quality of the bounds found

Instance	ULB		IULB	
	GAP	Time	Gap	Time
CETSP-25-01	16.17%	95.63	15.59%	95.24
CETSP-25-02	14.02%	16.64	13.69%	17.58
CETSP-25-03	14.34%	102.01	13.98%	92.04
CETSP-25-04	15.08%	21.19	14.64%	21.96
CETSP-25-05	14.91%	55.11	14.54%	46.81
CETSP-25-06	15.39%	93.91	14.89%	33.45
CETSP-25-07	14.06%	28.70	13.54%	61.53
CETSP-25-08	13.79%	57.22	13.13%	22.08
CETSP-25-09	15.41%	132.46	14.07%	61.20
CETSP-25-10	16.15%	49.67	16.04%	22.93
CETSP-30-01	17.89%	82.28	17.00%	98.13
CETSP-30-02	18.80%	78.37	17.56%	39.26
CETSP-30-03	15.86%	334.09	14.93%	240.33
CETSP-30-04	19.31%	866.01	18.19%	1005.15
CETSP-30-05	18.15%	129.34	17.59%	59.04
CETSP-30-06	16.78%	56.07	16.23%	79.40
CETSP-30-07	16.58%	537.38	16.28%	94.79
CETSP-30-08	16.23%	76.08	15.80%	93.93
CETSP-30-09	18.04%	331.81	17.47%	307.89
CETSP-30-10	17.69%	232.45	16.80%	95.57

Table 2. Computational results of ULB and IULB algorithms on the the larger instances with r=0.5.

by two algorithms decreases. However, once again IULB results more effective than ULB in all the instances with an improvement that ranges from 0.66% (CETSP-25-10) to 8.71%(CETSP-25-09). These further results definitively state that, whatever are the size of the instances and the radius considered, the new ideas implemented in IULB make this new algorithm always more effective than ULB. Surprisingly, the improvements concern even the performance. Indeed, IULB is faster than ULB in 13 out of 20 instances and in 8 cases its computational time is the half of the computational time of ULB. Finally, IULB solves 17 instances in less than 100 seconds while ULB, in the same time, solves only 12 instances.

7 Conclusions

In this article, we have presented an improved version of the adaptive internal discretization scheme and a heuristic that combines this scheme with a second-order cone programming algorithm. The computational results carried out on benchmark instances revealed that the new algorithm outperforms previous approaches to CETSP, in terms of quality of the bounds and, often, of the computational time. A possible direction for future work is to improve Algorithm 1 with the Carousel Greedy [8] or to develop new effective metaheuristics like

Tabu Search [4, 7] and Genetic Algorithm [3, 6] without applying discretization schemes.

References

1. Erling D Andersen, Cornelis Roos, and Tamas Terlaky. On implementing a primal-dual interior-point method for conic quadratic optimization. Mathematical Programming, 95(2):249–277, 2003.
2. B. Behdani and J.C. Smith. An integer-programming-based approach to the close-enough traveling salesman problem. INFORMS Journal on Computing, 26(3):415–432, 2014.
3. F. Carrabs, C. Cerrone, and R. Cerulli. A memetic algorithm for the weighted feedback vertex set problem. Networks, 64(4):339–356, 2014.
4. F. Carrabs, C. Cerrone, and R. Cerulli. A tabu search approach for the circle packing problem. In 2014 17th International Conference on Network-Based Information Systems, pages 165–171. IEEE, 2014.
5. F. Carrabs, C. Cerrone, R. Cerulli, and M. Gaudioso. A novel discretization scheme for the close enough traveling salesman problem. Computers & Operations Research, 78:163–171, 2017.
6. C. Cerrone, R. Cerulli, and M. Gaudioso. Omega one multi ethnic genetic approach. Optimization Letters, 10(2):309–324, 2016.
7. C Cerrone, R Cerulli, and M Gentili. Vehicle-id sensor location for route flow recognition: Models and algorithms. European Journal of Operational Research, 247(2):618–629, 2015.
8. C. Cerrone, R. Cerulli, and B. Golden. Carousel greedy: a generalized greedy algorithm with applications in optimization. University of Maryland (submitted for publication).
9. W.P. Coutinho, R.Q. Do Nascimento, A.A. Pessoa, and A. Subramanian. A branch-and-bound algorithm for the close-enough traveling salesman problem. INFORMS Journal on Computing, 28(4):752–765, 2016.
10. J. Dong, N. Yang, and M. Chen. Heuristic approaches for a tsp variant: The automatic meter reading shortest tour problem. Operations Research/ Computer Science Interfaces Series, 37:145–163, 2007.
11. M. Fischetti, J. Salazar-Gonzalez, and P. Toth. The generalized traveling salesman and orienteering problems. In The Traveling Salesman Problem and Its Variations, volume 12 of Combinatorial Optimization, pages 609–662. Springer US, 2007.
12. D. Gulczynski, J. Heath, and C. Price. Close enough traveling salesman problem: A discussion of several heuristics. In Perspectives in Operations Research, volume 36 of Operations Research/Computer Science Interfaces Series, pages 271–283. Springer US, 2006.
13. W.K. Mennell. Heuristics for solving three routing problems: Close-enough traveling salesman problem, close-enough vehi- cle routing problem, sequence-dependent team orienteering problem. PhD thesis, The Robert H. Smith School of Business, University of Maryland, College Park., 2009.
14. W.K. Mennell, B. Golden, and 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.
15. B. Yuan, M. Orlowska, and S. Sadiq. On the optimal robot routing problem in wireless sensor networks. IEEE Transactions on Knowledge and Data Engineering, 19(9):1252–1261, 2007.