# Exact and approximate formulations for the close-enough TSP

Domingo Araya[1], Gustavo Angulo[1†], Margarita Castro[1*†]

[1*]Department of Industrial and Systems Engineering, Pontificia Universidad Católica de Chile, Av. Vicuña Mackenna 4860, Santiago de Chile, 7820436, Chile.

*Corresponding author(s). E-mail(s): margarita.castro@uc.cl;
Contributing authors: domingo.araya@uc.cl; gangulo@uc.cl;
†These authors contributed equally to this work.

**Abstract**

This work addresses the Close-Enough Traveling Salesman Problem (CETSP), a variant of the classic traveling salesman problem in which we seek to visit neighborhoods of points in the plane (defined as disks) rather than specific points. We present two exact formulations for this problem based on second-order cone programming (SOCP), along with approximated mixed-integer linear formulations derived from polyhedral approximations of the second-order cone constraints. While SOCP models yield optimal solutions, they are quite inefficient for larger instances. Thus, we proposed several decomposition schemes based on exact SOCP models and their corresponding MILP approximations, and several cuts based on SOCP duality and classic integer L-shape methods. Moreover, we illustrate why Generalized Benders cuts cannot be applied to these problems and the consequences of doing so anyway. We test all the proposed procedures over a wide range of instances and analyze the results for different instance features (i.e., number of points and radius size). We note that exact decomposition methods based on the MILP approximation with a SOCP subproblem achieve the best overall performance, which illustrates the benefit of combining methodologies from the MILP and SOCP literature to address the problem.

**Keywords:** keyword1, Keyword2, Keyword3, Keyword4

## 1 Introduction

The Traveling Salesman Problem (TSP) is a classic combinatorial optimization problem that has been the subject of extensive study for several decades. Given a set of locations and pairwise distances, the goal is to find the shortest possible tour that starts and ends at a designated location while visiting every other location exactly once. The TSP has several applications in a variety of fields, such as logistics, genome sequencing, and data clustering [1].

This work addresses the Close-Enough Traveling Salesman Problem (CETSP), a generalization of the TSP in which it is sufficient to visit a previously defined compact region, known as its neighborhood, that contains the exact location. Although the problem can be defined for arbitrary compact regions, we assume that the neighborhoods are closed disks centered in the nodes, which is a common assumption in the CETSP literature [2, 3], and corresponds precisely to the type of neighborhoods that arise naturally in applications of the CETSP, which include: distant meter readings via radio frequency identification technology [4], UAV routing in various contexts [3], and robot monitoring of wireless sensor networks [5].

Several generalizations of the CETSP have been proposed to capture additional practical requirements. For instance, Di Placido et al. [6] propose the generalized CETSP, a variation of the problem where each location has several neighborhoods instead of a single one, with neighborhoods given by disks of different radii, and a prize associated with each region which is redeemed by traversing it. Carrabs et al. [7] propose the multiple CETSP, in which there are several vehicles, and the objective is to minimize the length of the longest tour assigned to a vehicle. Gao et al. [8] study a variant of the CETSP called the pickup-and-delivery TSP with neighborhoods (PDTSPN), which combines traditional pickup-and-delivery requirements with the flexibility of the CETSP. In this context, they propose a cut-generation scheme to solve the problem, based on generalized Benders decomposition [9].

While there are several heuristic approaches to solving the CETSP [4, 10, 11], there is a notorious scarcity of exact algorithms with optimality guarantees. Mennel [10] introduced the first mathematical formulation of the problem, along with a heuristic approach based on finding non-empty neighborhood intersections, to subsequently solve a TSP over a selection of points from said intersections. Additionally, they introduced the *overlap ratio* as a measure of instance difficulty, defined as the ratio of the mean radius to the largest side of the smallest rectangle containing all neighborhoods. Gentilini et al. [12] presented a solution approach for this formulation based on a traditional spatial branch-and-bound algorithm for mixed-integer non-linear programming (MINLP), with specific improvements for the CETSP to make the problem more tractable for MINLP solvers. Behdani and Smith [2] proposed a discretization scheme that generates lower and upper bounds on most test instances, and can be iteratively refined to make said bounds arbitrarily close. Building on these results, Carrabs et al. [13] introduced a novel discretization scheme that improves upon those of [2], generally providing better bounds in less time.

To the best of our knowledge, the only exact algorithm specific to this problem is a combinatorial *branch-and-bound* [3], where each node of the tree corresponds to a partial visit sequence, for which the visit points for each neighborhood are computed via second-order cone programming (SOCP). If the solution does not visit every neighborhood, the partial sequence is extended, creating new nodes of the branching tree. This approach provides good results on several benchmark instances but, in the worst case, has to consider every possible partial sequence. This algorithm was refined by Zhang et al. [14], who proposed a number of improvements that lead to better results in most instances of the problem. Gutow and Choset [15] further improved the algorithm by reusing information from parent nodes in the branching tree to speed up the solution of subproblems corresponding to the children nodes, while Deckerová et al. [16] extended it to a more general setting.

In this paper, we present two SOCP-based formulations for the CETSP, along with approximated formulations based on mixed-integer linear programming (MILP) that provide arbitrarily close approximations to the feasible regions of the original formulations. Furthermore, we present decomposition schemes for both the SOCP and MILP formulations in order to make the problem more computationally manageable and, in the case of the MILP models, to close the gap left by the approximation factor. We also provide examples that demonstrate that the cuts proposed by [8] are not always valid for the problem. Finally, we perform exhaustive computational experiments over a variety of instances to measure and compare the performance of the previously mentioned formulations.

The remainder of the paper is organized as follows. Section 2 contains a formal definition of the CETSP, along with the notation that is used throughout the paper for the parameters of the problem. Section 3 presents the SOCP-based exact formulations, while Section 4 introduces the linear approximation scheme and presents the resulting formulations. Section 5 describes the two-stage decomposition scheme. In Section 6, we argue that the cuts proposed by [8] are not valid, and we then present computational results in Section 7. Finally, conclusions are presented in Section 8.

## 2 Problem description

The CETSP can be formally stated as follows. Let $N = \{1, \dots, n\}$ be a set of locations in a two-dimensional plane, with their corresponding coordinates $c_i = (c_i^x, c_i^y)$ for each $i \in N$. Each location $i \in N$ is covered by a closed disk centered in $c_i$ with radius $r_i$, which we call its neighborhood. In this setting, the CETSP seeks the shortest tour that visits each neighborhood, with no restrictions on the order of visits. We consider a neighborhood to be visited if the intersection of the tour and the neighborhood is nonempty. Although the problem can be stated in terms of arbitrary norms, we consider Euclidean distances. We also assume $r_1 = 0$, so location 1 acts as the depot.

Behdani and Smith [2] proved that any optimal solution can be characterized by line segments between the neighborhoods. Hence, the problem reduces to finding a visit sequence along with the endpoints of the corresponding line segments, such that the tour over these points forms a Hamiltonian cycle of minimum length and each point is contained in its corresponding neighborhood. Figure 1 illustrates an optimal CETSP tour for an instance with 5 nodes, exemplifying this property of optimal CETSP solutions.

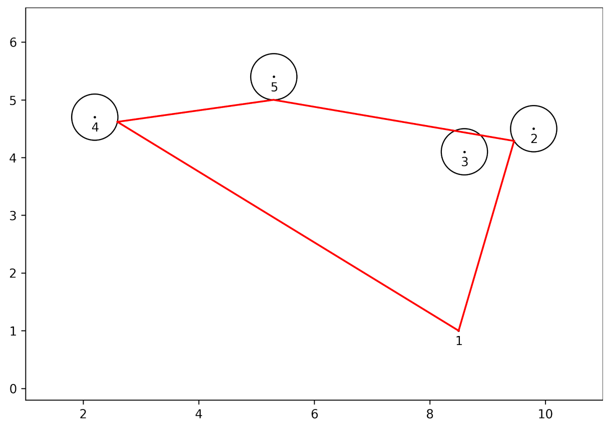

**Fig. 1**: An optimal CETSP tour with $|N| = 5$.

## 3 SOCP formulations

We now present our two SOCP formulations for the CETSP. The first model leverages classic TSP formulation over undirected graphs, while the second focuses on the sequential aspect of the problem. For notation purposes, we consider $A = \{(i, j) \in N \times N : i \neq j\}$ as the set of all location pairs that are distinct to each other and, similar to the classic TSP, we refer to each $(i, j) \in A$ as arcs.

Our first SOCP formulation is a natural generalization of the classic TSP formulation with a binary decision variable for each arc, which we call *arc-based formulation* (ABF). Here, variables $x_{ij} \in \{0, 1\}$ indicate whether the tour goes from the neighborhood of location $i \in N$ to the neighborhood of location $j \in N$ or not. Variables $u_{ij} \geq 0$ are auxiliary variables utilized for subtour elimination, and variables $p_i \in \mathbb{R}^2$ represent the coordinates of the chosen point for the neighborhood corresponding to each location $i \in N$. Finally, variables $d_{ij} \geq 0$ represent the distance between the points chosen in each neighborhood, and variables $d^a_{ij} \geq 0$ are auxiliary variables that represent the distance of the line segments that compose the tour, taking the value 0 if the tour does not go from $i$ to $j$. Then, the ABF formulation is as follows:

$$\min \sum_{(i,j)\in A} d^a_{ij} \tag{ABF}$$

$$\text{s.t.} \sum_{j\in N} x_{ij} = 1, \qquad \forall i \in N, \tag{1a}$$

$$\sum_{i\in N} x_{ij} = 1, \qquad \forall j \in N, \tag{1b}$$

$$\sum_{j\in N} u_{ij} - \sum_{j\in N} u_{ji} = -1, \qquad \forall i \in N \setminus \{1\}, \tag{1c}$$

$$\sum_{j\in N} u_{1j} - \sum_{j\in N} u_{j1} = n - 1, \tag{1d}$$

$$u_{ij} \leq (n-1)x_{ij}, \qquad \forall (i,j) \in A, \tag{1e}$$
$$u_{ij} \geq 0, \qquad \forall (i,j) \in A, \tag{1f}$$
$$||p_i - c_i|| \leq r_i, \qquad \forall i \in N, \tag{1g}$$
$$||p_i - p_j|| \leq d_{ij}, \qquad \forall (i,j) \in A, \tag{1h}$$
$$d_{ij} - M_{ij}(1 - x_{ij}) \leq d^a_{ij}, \qquad \forall (i,j) \in A, \tag{1i}$$
$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A, \tag{1j}$$
$$d_{ij}, d^a_{ij} \geq 0, \qquad \forall (i,j) \in A, \tag{1k}$$
$$p_i \in \mathbb{R}^2, \qquad \forall i \in N. \tag{1l}$$

The objective function of ABF minimizes the total distance of the tour. Constraints (1a) and (1b) are degree constraints, constraints (1c) to (1f) are subtour elimination constraints [17], constraint (1g) ensures that the points picked are contained in the neighborhoods, constraint (1h) defines the distance between representative points, and constraint (1i) activates the variables $d^a_{ij}$, causing them to be equal to $x_{ij}d_{ij}$. Here, the parameter $M_{ij}$ corresponds to the maximum possible distance between neighborhoods $i$ and $j$, causing these constraints to be trivially satisfied if $x_{ij} = 0$. Lastly, constraints (1f), (1j)-(1l) define the nature of the variables.

A potential drawback of ABF is that it contains $\mathcal{O}(n^2)$ quadratic constraints, opening the possibility of the complexity rendering the problem intractable as $n$ increases. In addition, it relies on big-M constraints, which are known to have poor relaxations and to lead to numerical issues [18]. This motivates our second formulation, based on determining which neighborhood gets visited at each stage of the tour. We refer to this formulation as *sequence-based formulation* (SBF). In this formulation, variables $z_{ik} \in \{0,1\}$ indicate whether location $i \in N$ is visited in stage $k \in N$, $q_k \in \mathbb{R}^2$ represents the coordinates of the point visited in stage $k$, and $t_k \geq 0$ indicates the distance traveled in stage $k$. The formulation is as follows:

$$\min \sum_{k \in N} t_k \tag{SBF}$$
$$\text{s.t.} \sum_{k \in N} z_{ik} = 1, \qquad \forall i \in N, \tag{2a}$$
$$\sum_{i \in N} z_{ik} = 1, \qquad \forall k \in N, \tag{2b}$$
$$z_{11} = 1, \tag{2c}$$
$$||q_k - q_{k+1}|| \leq t_k, \qquad \forall k \in N, \tag{2d}$$
$$\left|\left|\sum_{i \in N} z_{ik}c_i - q_k\right|\right| \leq \sum_{i \in N} z_{ik}r_i, \qquad \forall k \in N, \tag{2e}$$
$$z_{ik} \in \{0,1\}, \qquad \forall (i,k) \in A, \tag{2f}$$
$$q_k \in \mathbb{R}^2,\ t_k \geq 0, \qquad \forall k \in N. \tag{2g}$$

The objective function of SBF minimizes the total distance traveled, constraint (2a) ensures that each neighborhood is visited at some stage, while constraint (2b) makes it so that one neighborhood is visited at each stage. Constraint (2c) fixes the first neighborhood to be visited, in order to eliminate symmetrical solutions given by shifting the sequence, constraint (2d) defines the distance between representative points (taking $n+1$ to be equal to 1), and constraint (2e) ensures that the representative point of stage $k$ is contained in the neighborhood visited in that stage. Finally, constraints (2f)-(2g) state the nature of the variables.

## 4 Approximated MILP formulations

Given the inherent complexity associated with solving the previous problems due to their second-order cone constraints and the superior performance of state-of-the-art MILP, it might be useful to approximate these conic constraints with polyhedral representations. For this purpose, we rely on the work of Ben-Tal and Nemirovski [19], who propose a linear approximation of conic constraints such that the resulting feasible region is arbitrarily close to the original feasible region, with a limited number of variables and constraints.

Formally, let $\epsilon > 0$ and $\mathbf{L}^2 = \{(y, w) \in \mathbb{R}^2 \times \mathbb{R} \mid w \geq ||y||\}$ be the 3-dimensional Lorentz cone. We say that a polyhedron $P_\epsilon \subseteq \mathbb{R}^2 \times \mathbb{R}$ is an $\epsilon$-approximation of $\mathbf{L}^2$ if every point $(y, w) \in P_\epsilon$ is such that $||y|| \leq (1+\epsilon)w$ . In this context, Ben-Tal and Nemirovski [19] show that $P_\epsilon(\nu)$ is an $\epsilon(\nu)$-approximation of $\mathbf{L}^2$, where $\nu \in \mathbb{Z}_+$ is a construction parameter that determines the precision, $\epsilon(\nu)$ is given by

$$\epsilon(\nu) = \left(\cos\left(\frac{\pi}{2^{\nu+1}}\right)\right)^{-1} - 1,$$

and the polyhedron $P_{\epsilon(\nu)}$ is defined as:

$$
\begin{aligned}
P_\epsilon(\nu) := \Bigg\{ &(y, w) \in \mathbb{R}^2 \times \mathbb{R} : \\
&\xi^0 \geq |y_1|, && && \text{(3a)}\\
&\eta^0 \geq |y_2|, && && \text{(3b)}\\
&\xi^j = \cos\left(\frac{\pi}{2^{j+1}}\right)\xi^{j-1} + \sin\left(\frac{\pi}{2^{j+1}}\right)\eta^{j-1}, && \forall j \in \{1, \dots, \nu\}, && \text{(3c)}\\
&\eta^j \geq \left|-\sin\left(\frac{\pi}{2^{j+1}}\right)\xi^{j-1} + \cos\left(\frac{\pi}{2^{j+1}}\right)\eta^{j-1}\right|, && \forall j \in \{1, \dots, \nu\}, && \text{(3d)}\\
&\xi^\nu \leq w, && && \text{(3e)}\\
&\eta^\nu \leq \tan\left(\frac{\pi}{2^{\nu+1}}\right)\xi^\nu, && && \text{(3f)}\\
&\xi^j, \eta^j \in \mathbb{R}, && \forall j \in \{0, \dots, \nu\}\Bigg\}. && \text{(3g)}
\end{aligned}
$$

Thus, the approximation is given by the projection onto the $(y, w)$-space of the polyhedron defined by the system of inequalities (3a)-(3g). We note that the number of constraints and new variables grows linearly with $\nu$; in particular, the system has $2(\nu+1)$ additional variables and $2(\nu+2)$ constraints. Moreover, small values of $\nu$ already yield close approximations (e.g., setting $\nu = 3$ yields $\epsilon(\nu) < 0.02$), thus the resulting polyhedron $P_{\epsilon(\nu)}$ remains manageable in terms of dimensionality.

Figure 2 illustrates the evolution of the approximation as $\nu$ increases, showing cross-sections of the approximation $P_{\epsilon(\nu)}$ and of the Lorentz cone on $w = 1$, for different values of $\nu$. In each subfigure, the inner disk corresponds to the cross-section of the Lorentz cone, while the outer polygon corresponds to the approximation $P_{\epsilon(\nu)}$. This figure clearly shows that the approximation approaches the Lorentz cone as $\nu$ increases, and that it is already quite close for small $\nu$.

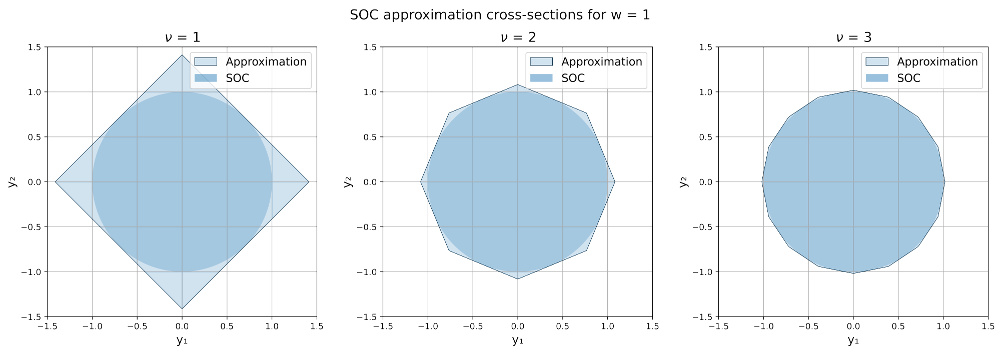

**Fig. 2**: Cross-sections of the Lorentz cone and its approximation for different values of $\nu$

We use this approximation to transform our SOCP models into MILPs without any conic constraints. In the arc case, the MILP approximation (ABF-A) is:

$$\min \sum_{(i,j)\in A} d^a_{ij} \tag{ABF-A}$$

$$\text{s.t. } (1a) - (1f), (1i) - (1l),$$

$$(p_i - c_i, r_i) \in P_{\epsilon(\nu)}, \qquad \forall i \in N, \tag{4a}$$

$$(p_i - p_j, d_{ij}) \in P_{\epsilon(\nu)}, \qquad \forall (i,j) \in A. \tag{4b}$$

The objective function is the same as in the original formulation, the first constraint group states the unchanged constraints from the original model, and (4a) along with (4b) ensure that the solution lies in the approximation of the starting conic constraints.

Similarly, we apply this approximation to obtain a MILP formulation (SBF-A) for the sequence case:

$$\min \sum_{k\in N} t_k \tag{SBF-A}$$

$$\text{s.t. } (2a) - (2c), (2f) - (2g),$$

$$(q_k - q_{k+1}, t_k) \in P_{\epsilon(\nu)}, \qquad \forall k \in N, \tag{5a}$$

$$\left( \sum_{i \in N} z_{ik} c_i - q_k, \sum_{i \in N} z_{ik} r_i \right) \in P_{\epsilon(\nu)}, \qquad \forall k \in N. \tag{5b}$$

As in the previous formulation, the objective function remains unchanged from SBF, the first constraint group states the non-conic constraints, and (5a) with (5b) force the solution to be in the polyhedral approximation of the original feasible region induced by the conic constraints.

Note that the feasible regions obtained by these approximations are slightly larger than those of the original problem because we use an outer approximation for each neighborhood. Thus, models ABF-A and SBF-A provide lower bounds for the original problem, and as such, the optimal solutions obtained by such models might be infeasible for the original problem. To obtain a feasible solution (and thus an upper bound), one can take the optimal visit sequence of either ABF-A or SBF-A, fix the binary variables in the original formulation, and solve for the remaining variables using second-order cone programming. Indeed, we use a similar idea to obtain optimal solutions from these MILP approximations, which we further detail in the following section.

## 5 Decomposition schemes

We now present decomposition schemes for all of our previously presented formulations. In the case of SOCP formulations, these decompositions aim to reduce the complexity inherent in these models, whereas in the case of MILP approximations, they aim to close the optimality gap left by the approximation factor.

All four decompositions have the following structure: (i) a master problem that is a relaxation of the original problem and, as such, estimates the distance traveled; (ii) a subproblem that computes the actual optimal distance for the visit sequence obtained in the master problem. If the estimation obtained in the master problem differs from the subproblem value, a new cut is generated and added to the master problem. This procedure is iterated until the values obtained in the master problem and the subproblem match for some solution.

### 5.1 Arc-based decomposition

First, we present the decomposition approach based on the ABF model (i.e., ABF-D). Here, the master problem retains all tour-related variables and constraints, using underestimations of the distance between neighborhoods as weights in the objective function, while the subproblem takes the visit sequence given by the master problem and computes the difference between the estimation and the actual optimal tour length for that sequence. Then, the master problem is as follows:

$$\min \sum_{(i,j) \in A} x_{ij} m_{ij} + \theta \tag{ABF-D}$$

$$\text{s.t. } (1a) - (1f), (1j),$$

$$\theta \geq 0, \tag{6a}$$
$$\theta \geq a^T x + b, \qquad \forall (a,b) \in C, \tag{6b}$$

where $m_{ij}$ is an underestimator of the distance between neighborhoods $i$ and $j$, for which we use the minimum possible distance between the said neighborhoods. The new variable $\theta$ represents the difference between the estimation of the total distance and the actual distance for a given visit sequence. Set $C$ is originally empty and contains the parameters of the added cuts to the master problem. In this context, the first constraint group (i.e., (1a)-(1f) and (1j)) guarantees that $x$ induces a valid tour, while (6a) sets a starting lower bound for $\theta$ (since we are underestimating the tour distance, this difference will always be non-negative, thus making 0 a valid lower bound), and (6b) ensures that the value of $\theta$ does not underestimate the difference between the estimated total distance and the actual distance.

The subproblem is as follows:

$$Q^A(x) = \min \sum_{(i,j)\in S(x)} (d_{ij} - m_{ij}) \tag{7a}$$
$$\text{s.t. } ||p_i - c_i|| \leq r_i, \qquad \forall i \in N, \tag{7b}$$
$$||p_i - p_j|| \leq d_{ij}, \qquad \forall (i,j) \in S(x), \tag{7c}$$
$$d_{ij} \geq 0, \qquad \forall (i,j) \in S(x), \tag{7d}$$
$$p_i \in \mathbb{R}^2, \qquad \forall i \in N. \tag{7e}$$

Here, set $S(x) = \{(i,j) \in A \mid x_{ij} = 1\}$ is the support of $x$, and, as such, the constraints ensure that the points selected for the visit sequence $x$ satisfy neighborhood constraints and properly activate the distance variables. Note that, since we instantiate the subproblem over $S(x)$, which always has cardinality $n$, the subproblem has $2n$ conic constraints, thus, a considerably lesser amount than the original $n^2 + n$ conic constraints in ABF.

In order to solve this decomposition, we solve the master problem and, given an integer solution $\hat{x}$, we compute $Q^A(\hat{x})$. If the corresponding master solution $\hat{\theta}$ is such that $\hat{\theta} < Q^A(\hat{x})$, then we add the following optimality cuts to the master problem [20]:

$$-\frac{1}{2}Q^A(\hat{x}) \left( \sum_{(i,j)\in S(\hat{x})} (1 - x_{ij}) \right) + Q^A(\hat{x}) \leq \theta, \tag{8a}$$
$$-\frac{1}{2}Q^A(\hat{x}) \left( \sum_{(i,j)\in S(\hat{x})} (1 - x_{ji}) \right) + Q^A(\hat{x}) \leq \theta. \tag{8b}$$

Note that (8a) exclusively eliminates the solution $\left(\hat{x}, \hat{\theta}\right)$, since, for any feasible $x$ different from $\hat{x}$, they will differ in at least two entries, thus leading the cut to being dominated by constraint (6a), and, as such, trivially satisfied. The cut (8b) does the same for the visit sequence given by reversing the arcs utilized, which will always induce the same total distance due to the symmetric nature of the problem. Note that these cuts are enumerative in nature and, as such, we could potentially need to add a cut for every possible tour.

## 5.2 Arc-based decomposition with MILP approximation

Given that ABF-D considers just a naive approximation of the real distance between neighborhoods, we now present a decomposition based on ABF-A (i.e., ABF-A-D) to strengthen the master problem. By doing so, we aim to avoid the need to enumerate all possible sequences in the master problem to find the optimal solution. In this decomposition, the master problem consists of ABF-A with an added variable that estimates the difference between the distance obtained by the approximation and the actual distance, while the subproblem computes the value of said difference for a given visit sequence.

The master problem is given by ABF-A-D, where the meaning of variable $\theta$, set $C$ and constraints (9b) and (9b) are the same as in ABF-D.

$$\begin{aligned} \min\ & \sum_{i\in N}\sum_{j\in N} d^a_{ij} + \theta && \text{(ABF-A-D)} \\ \text{s.t. }& (1a)-(1f), (1i)-(1l), && \\ & (4a)-(4b), && \\ & \theta \geq 0, && (9a) \\ & \theta \geq a^T x + b, \qquad \forall (a,b)\in C. && (9b) \end{aligned}$$

The subproblem is as follows:

$$Q^A_\nu(x, d^a) = \min \left\{ \sum_{(i,j)\in S(x)} \left(d_{ij} - d^a_{ij}\right) : (7b) - (7e) \right\}, \qquad (10)$$

which is analogous to the subproblem of ABF-D and only changes the objective function to adapt to the estimation $d^a$ returned by the master problem. Similar to the previous the decomposition, we solve the master problem and, once a solution $(\hat{x}, \hat{d^a}, \hat{\theta})$ is found, we solve $Q^A_\nu(\hat{x}, \hat{d^a})$ and get its optimal value. If $\hat{\theta} < Q^A_\nu(\hat{x}, \hat{d^a})$, we add cuts (8a) and (8b), replacing $Q^A(\hat{x})$ by $Q^A_\nu(\hat{x}, \hat{d^a})$.

## 5.3 Sequence-based decomposition

We also consider decomposition methodologies based on the SBF model (i.e., SBF-D). The master problem in this decomposition determines which neighborhood to visit at each stage, while the subproblem computes the distance associated with that visit order, that is,

$$\begin{aligned} \min\ & \theta && \text{(SBF-D)} \\ \text{s.t. }& (2a)-(2c), (2f), && \end{aligned}$$

$$\theta \geq 0, \tag{11a}$$
$$\theta \geq a^T z + b, \qquad \forall (a,b) \in C, \tag{11b}$$

Here, the variable $\theta$ estimates the total distance of the tour determined by variable $z$, with the first group of constraints ensuring that said tour is valid, while (11a) and (11b) work just as in the previous decompositions. Its corresponding subproblem is:

$$Q^S(z) = \min \sum_{k \in N} t_k \tag{12a}$$
$$\text{s.t. } ||q_k - q_{k+1}|| \leq t_k, \qquad \forall k \in N, \tag{12b}$$
$$\left|\left|\sum_{i \in N} z_{ik} c_i - q_k\right|\right| \leq \sum_{i \in N} z_{ik} r_i, \qquad \forall k \in N, \tag{12c}$$
$$q_k \in \mathbb{R}^2, t_k \geq 0, \qquad \forall k \in N. \tag{12d}$$

The constraints in this subproblem are identical to those that determine the visit points and the distances in SBF, thus, it computes the optimal distance for the visit sequence encoded a given $z$.

Given that the objective function of the master problem is simply $\theta$, employing the enumerative cuts in ABF-D would force us to effectively iterate over every single possible sequence, rapidly rendering them ineffective as $n$ grows. In order to deal with this issue, use duality theory over $Q^S(z)$ to obtain stronger cuts as needed. Note that the subproblem satisfies Slater's condition and is bounded from below (by 0), so strong duality holds [21]. Taking the dual, we can re-formulate $Q^S(z)$ as follows:

$$Q^S(z) = \max \sum_{i \in N} \sum_{k \in N} \mu_{ik} z_{ik} \tag{13a}$$
$$\text{s.t. } 1 - \lambda_k^n - \lambda_k^d = 0, \qquad \forall k \in N, \tag{13b}$$
$$\mu_{ik} + c_i \cdot \rho_k^c + r_i \lambda_k^c = 0, \qquad \forall (i,k) \in A, \tag{13c}$$
$$\rho_k^d - \rho_{k-1}^d + \rho_k^c = 0, \qquad \forall k \in N, \tag{13d}$$
$$||\rho_k^d|| \leq \lambda_k^d, \qquad \forall k \in N, \tag{13e}$$
$$||\rho_k^c|| \leq \lambda_k^c, \qquad \forall k \in N, \tag{13f}$$
$$\lambda_k^n, \lambda_k^d, \lambda_k^c \geq 0, \qquad \forall k \in N, \tag{13g}$$
$$\mu_{ik} \in \mathbb{R}, \qquad \forall (i,k) \in A, \tag{13h}$$
$$\rho_k^d, \rho_k^c \in \mathbb{R}^2, \qquad \forall k \in N. \tag{13i}$$

Variables with superscript $d$ are dual variables associated with distance constraints (12b), variables with superscript $c$ are those associated with distance to the center constraints (12c), and variables with superscript $n$ are associated with variable nature constraints (12d). For brevity, we assume in constraint (13d) that $k - 1 = n$ when $k = 1$. Finally, $\mu$ is associated with an auxiliary constraint utilized to build the dual problem, and can in fact be eliminated, since it is fully determined by constraint (13c), so we can integrate this constraint into the objective function, eliminating the variable $\mu$. However, for notational simplicity, we retain this variable in the model.

Now, given an integer solution $\hat{z}$ to the master problem, we compute $Q^S(\hat{z})$, and, if $\hat{\theta} < Q^S(\hat{z})$, we retrieve the solution $\hat{\mu}$ of the subproblem and add the following optimality cut:

$$\sum_{i \in N} \sum_{k \in N} \hat{\mu}_{ik} z_{ik} \leq \theta \tag{14}$$

**Remark.** *We could use this duality approach to generate cuts for ABF-D, however, we opt not to because variable $x$ implicitly appears in $Q^A(x)$ when defining set $S(x)$ to simplify the subproblem. Thus, to obtain similar cuts for ABF-D, we need to reformulate $Q^A(x)$ by instantiating over $A$ instead of $S(x)$ and carrying over the big-M constraint (1i) to make $x$ explicit, thus considerably increasing the dimension of the model. Additionally, the cut obtained involves the big-M parameter and, in our preliminary experiments, this led to worse results than simply utilizing the proposed cuts (8a) and (8b).*

## 5.4 Sequence-based decomposition with MILP approximation

Following the same principle as in ABF-A-D, we enhance the sequence-based decomposition by introducing the MILP approximating constraints (i.e., SBF-A-D). The master problem corresponds to the SBF-A with an additional variable $\theta$ and its corresponding constraints, resulting in:

$$\begin{aligned}
\min \ & \sum_{k \in N} t_k + \theta && && \text{(SBF-A-D)} \\
\text{s.t. } & (2a) - (2c), (2f) - (2g) \\
& (5a) - (5b) \\
& \theta \geq 0 && && \text{(15a)} \\
& \theta \geq a^T z + b && \forall (a,b) \in C && \text{(15b)}
\end{aligned}$$

We define the subproblem of this decomposition as $Q^S_\nu(\hat{z}, \hat{t}) = Q^S(\hat{z}) - \sum_{k \in N} \hat{t}_k$. We can use either the primal or the dual version of the subproblem in SBF-D (i.e., $Q^S(\hat{z})$) to generate the cuts. In both cases, if we get a solution such that $\hat{\theta} < Q^S_\nu(\hat{z})$, we add the following cuts, analogous to the ones used in ABF-D:

$$-\frac{1}{2} Q^S_\nu(\hat{z}) \left( \sum_{i,k \in S(\hat{z})} (1 - z_{ik}) \right) + Q^S_\nu(\hat{z}) \leq \theta, \tag{16a}$$

$$-\frac{1}{2} Q^S_\nu(\hat{z}) \left( \sum_{i,k \in S(\hat{z})} (1 - z_{i(n-k+1)}) \right) + Q^S_\nu(\hat{z}) \leq \theta. \tag{16b}$$

Here, (16a) works just as (8a), and (16b) is the cut corresponding to the sequence obtained by reversing the visit order described by $\hat{z}$, which has the same optimal tour distance, enabling us to include this cut. Alternatively, if using the dual version of $Q^S(\hat{z})$, we can add the following cut:

$$\sum_{i\in N}\sum_{k\in N}\hat{\mu}_{ik}z_{ik} - \sum_{k\in N}\hat{t}_k \leq \theta. \tag{17}$$

This cut is analogous to the one used in SBF-D, just adapting it to account for the additional constant term in the subproblem's objective function.

## 6 On the applicability of generalized Benders

The decompositions presented in the previous section utilized cuts that leverage SOCP duality or the binary nature of the master problem variables (i.e., enumerative cuts). An alternative is to consider Generalized Benders Decomposition (GBD) [9], which was proposed to solve the PDTSPN problem [8], a variant of the CETSP. In what follows, we explain some of the basic concepts regarding GBD and show that GBD, as proposed by Gao et al. [8] for the PDTSPN, cannot be applied to the CETSP (and the PDTSPN) without losing optimality guarantees.

As its name suggests, GBD was proposed as an extension of the classic Benders decomposition [22] to handle non-linear subproblems, allowing the decomposition of problems in which the subproblem is a convex optimization problem, with applications to some non-convex problems as well [23]. The approach considers problems of the following form:

$$\min_{x,y}\{f(x,y):\ G(x,y)\leq 0, x\in X, y\in Y\} \tag{18}$$

where $X$ and $Y$ are real-valued domains, $f: X\times Y\to\mathbb{R}$ is the objective function, and $G: X\times Y\to\mathbb{R}^m$ is a vector of constraint functions. This problem can be re-stated as

$$\min_{x}\{v(x):\ x\in X\cap V\}, \tag{19}$$

where $v(x)=\inf_{y\in Y}\{f(x,y) \text{ s.t. } G(x,y)\leq 0\}$ is the optimal value of the problem for a fixed $x$, and $V=\{x\in X: G(x,y)\leq 0 \text{ for some } y\in Y\}$ is the projection of the feasible region of the original problem onto the domain of variables $x$, i.e., the set of all $x\in X$ such that there exists a feasible value assignment for $y$. Under some fairly weak assumptions on $f$, $G$, and $Y$ [9], it holds that

$$v(x)=\sup_{u\geq 0}\left[\inf_{y\in Y}\left\{f(x,y)+u^\top G(x,y)\right\}\right]. \tag{20}$$

With this result, the original problem can be reformulated as

$$\begin{aligned}
\min_{x\in X}\quad & \theta && \text{(21a)}\\
\text{s.t.}\quad & \theta\geq\inf_{y\in Y}\left\{f(x,y)+u^\top G(x,y)\right\}, \quad \forall u\geq 0 && \text{(21b)}\\
& x\in V && \text{(21c)}
\end{aligned}$$

The logic of the decomposition is to relax constraints (21b) and (21c) to obtain a master problem in variable $x$. Given a solution $\hat{x}$ of the master problem, the approach iteratively solves the subproblem $v(\hat{x})$ and adds optimality cuts (from (21b)) or feasibility cuts (from (21c)) as needed until convergence. In the case of the CETSP, the subproblem is always feasible, so we focus solely on optimality cuts, which take the form

$$L^*(x, \hat{u}) \leq \theta, \tag{22}$$

where

$$L^*(x, \hat{u}) = \inf_{y \in Y} \left\{ f(x, y) + \hat{u}^\top G(x, y) \right\}, \tag{23}$$

and $\hat{u}$ is the optimal dual vector associated with the subproblem $v(\hat{x})$.

An important remark regarding the viability of GBD is that, in order for the cuts (22) to be practical, it is necessary that the function $L^*(\cdot, \hat{u})$ on $X$ can be obtained explicitly in a way that requires little or no more effort than that which is required to evaluate it at a single value of $x$, which the author of [9] refers to as Property (P). Otherwise, the functional form of the cuts would be too complex to be of any practical use. An example where Property (P) holds is when both $f$ and $G$ are linearly separable in $x$ and $y$.

Bagajewicz and Manousiouthakis [24] show that, if we assume that for each $\hat{u} \geq 0$, there exists some $\hat{y} \in Y$ such that the cuts can be expressed as

$$L^*(x, \hat{u}) = \inf_{y \in Y} \left\{ f(x, y) + \hat{u}^\top G(x, y) \right\} = f(x, \hat{y}) + \hat{u}^\top G(x, \hat{y}), \tag{24}$$

which, as a condition, is stronger than Property (P), then it is possible for the GBD procedure to converge to a non-optimal solution that can, in fact, not even be a local minimum.

In the context of the PDTSPN, Gao et al. [8] propose a formulation similar to ABF, for which they implement a decomposition scheme based on GBD. The resulting master problem in their decomposition is the same as ABF-D (with additional constraints corresponding to the pickup-and-delivery aspect of the problem), and the subproblem is as follows:

$$\begin{aligned}
Q(x) = \min \quad & \sum_{(i,j) \in A} x_{ij}(d_{ij} - m_{ij}) && && \text{(25a)}\\
\text{s.t.} \quad & f_i(p_i) \leq 0 && \forall i \in N && \text{(25b)}\\
& ||p_i - p_j|| \leq d_{ij} && \forall (i,j) \in A && \text{(25c)}\\
& d_{ij} \geq 0 && \forall (i,j) \in A && \text{(25d)}\\
& p_i \in \mathbb{R}^2 && \forall i \in N && \text{(25e)}
\end{aligned}$$

where functions $f_i$ represent the neighborhoods, which are assumed to be convex and compact. Note that circular neighborhoods are a special case of these functions.

This subproblem differs from the one we present in ABF-D in that it is instantiated over all arcs in $A$, instead of only over the arcs in the support of $x$, $S(x)$.

For simplicity, we assume that the distance estimations $m_{ij}$ are all equal to 0. In this context, cuts (22) take the following form:

$$\underbrace{\inf_{(p,d)\in P}\left\{\sum_{(i,j)\in A} d_{ij}x_{ij} - \sum_{i\in N}\hat{\pi}_i f_i(p_i) - \sum_{(i,j)\in A}\hat{\mu}_{ij}(||p_i - p_j|| - d_{ij})\right\}}_{L^*(x,\hat{\pi},\hat{\mu})} \leq \theta \tag{26}$$

where

$$\begin{aligned} P = \{(p,d) : ||p_i - p_j|| \leq d_{ij},\ 0 \leq d_{ij} & \qquad \forall (i,j) \in A, \\ f_i(p_i) \leq 0,\ p_i \in \mathbb{R}^2 & \qquad \forall i \in N\} \end{aligned}$$

is the feasible region of the subproblem (note that it does not depend on $x$), and $\hat{\pi}$ and $\hat{\mu}$ are the optimal dual vectors associated with the neighborhood constraints (25b) and the distance constraints (25c), respectively.

Based on this formulation, Gao et al. [8] propose that, given a master problem solution $\hat{x}$, and the corresponding primal and dual optimal solutions of the subproblem $(\hat{p}, \hat{d})$ and $(\hat{\pi}, \hat{\mu})$, respectively, the following cut can be added to the master problem if needed:

$$\sum_{(i,j)\in A} \hat{d}_{ij}x_{ij} - \sum_{i\in N}\hat{\pi}_i f_i(\hat{p}_i) - \sum_{(i,j)\in A}\hat{\mu}_{ij}(||\hat{p}_i - \hat{p}_j|| - \hat{d}_{ij}) \leq \theta. \tag{27}$$

Since $x$ does not appear in the second two summations, by complementary slackness, these terms are equal to zero, thus simplifying the cut to

$$\sum_{i,j\in N, i\neq j} \hat{d}_{ij}x_{ij} \leq \theta \tag{28}$$

However, this cut is not valid, since

$$L^*(x,\hat{\pi},\hat{\mu}) \leq \sum_{(i,j)\in A} \hat{d}_{ij}x_{ij} - \sum_{i\in N}\hat{\pi}_i f_i(\hat{p}_i) - \sum_{(i,j)\in A}\hat{\mu}_{ij}(||\hat{p}_i - \hat{p}_j|| - \hat{d}_{ij}), \tag{29}$$

as $(\hat{p}, \hat{d})$ is a feasible solution to the problem defining $L^*(x,\hat{\pi},\hat{\mu})$, but not necessarily optimal. Thus, the cuts proposed are overestimating $L^*(x,\hat{\pi},\hat{\mu})$, and, as such, are not valid. According to Bagajewicz and Manousiouthakis [24], using these cuts can lead to a procedure that converges to non-optimal solutions.

Specifically, we now show that such cuts are indeed invalid by analyzing their structure and using a numerical example. First, let us consider the sequence $\tilde{S}$ associated with some first-stage solution $\tilde{x}$ (i.e. $\tilde{S} = \{(i,j) \in A \mid \tilde{x}_{ij} = 1\}$). Given that $x$ is binary, the objective function of the subproblem can be rewritten as

$$\sum_{(i,j)\in A} d_{ij}\tilde{x}_{ij} = \sum_{(i,j)\in \tilde{S}} d_{ij}. \tag{30}$$

Let us note by $\tilde{d}$ the corresponding vector of optimal distances obtained by solving the subproblem. Note that, since only entries $(i, j) \in \tilde{S}$ appear in the objective function, and there is no upper bound on any of the distance variables, the optimal distance $\tilde{d}_{ij}$ for $(i, j) \notin \tilde{S}$ can take any non-negative value greater or equal to $||p_i - p_j||$, without affecting the optimality of the solution. Thus, the entries $\tilde{d}_{ij}$ for $(i, j) \notin \tilde{S}$ are not necessarily defined as the distance between the corresponding visit points, and can in fact be arbitrarily large, potentially leading to cuts that incorrectly eliminate feasible solutions.

We now present a numerical example that illustrates how this interpretation of (22) can cut off feasible solutions. Consider an instance with four nodes, whose coordinates and radii are given in Table 1, and plotted in Figure 3.

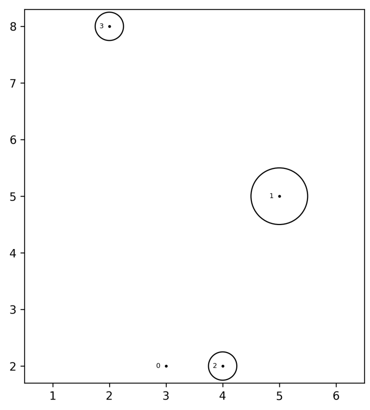

**Fig. 3**: Plot of the example instance

| Node | x | y | r |
|---|---|---|---|
| 0 | 3 | 2 | 0 |
| 1 | 5 | 5 | 0.5 |
| 2 | 4 | 2 | 0.25 |
| 3 | 2 | 8 | 0.25 |

**Table 1**: Coordinates and radii

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0.00 | 3.11 | 1.03 | 5.83 |
| 1 | 3.11 | 0.00 | 2.43 | 4.20 |
| 2 | 1.03 | 2.43 | 0.00 | 5.84 |
| 3 | 5.83 | 4.20 | 5.84 | 0.00 |

**Table 2**: Distance matrix for $0 \to 1 \to 2 \to 3 \to 0$

The optimal solution to this instance has a total distance of approximately 13.24, and the optimal visit order is $0 \to 3 \to 1 \to 2 \to 0$ (or the reverse). If we solve the subproblem for the visit order $0 \to 1 \to 2 \to 3 \to 0$, we can compute the distances manually given the optimal visit points to avoid the problem mentioned before, thus obtaining the distance matrix presented in Table 2.

Let us denote the optimal sequence by $S^*$, the second sequence by $\tilde{S}$, and its corresponding previously presented distance matrix by $\tilde{d}$. Suppose we generated the cut associated with $\tilde{S}$:

$$\sum_{(i,j)\in A} x_{ij}\tilde{d}_{ij} \leq \theta \tag{31}$$

which, given the distance matrix in Table 2, can be rewritten approximately as

$$\begin{aligned}&3.11(x_{01} + x_{10}) + 1.03(x_{02} + x_{20}) + 5.83(x_{03} + x_{30})\\&+ 2.43(x_{12} + x_{21}) + 4.20(x_{13} + x_{31}) + 5.84(x_{23} + x_{32}) \leq \theta\end{aligned} \tag{32}$$

Now, if we evaluate the left-hand side of this cut for sequence $S^*$, we obtain

$$\begin{aligned}\sum_{(i,j)\in S^*} \tilde{d}_{ij} &= \tilde{d}_{03} + \tilde{d}_{31} + \tilde{d}_{12} + \tilde{d}_{20}\\&\approx 5.83 + 4.20 + 2.43 + 1.03 = 13.49\end{aligned}$$

which is greater than the optimal distance of approximately 13.24.

Thus, the cut obtained by solving the subproblem for sequence $\tilde{S}$ leads to an overestimation of the distance of the optimal sequence $S^*$, and is therefore incorrect. This example illustrates that the cuts proposed by Gao et al. [8] are invalid, and, as such, the decomposition scheme they propose does not guarantee convergence to the optimal solution of the CETSP, or its extension, the PDTSPN.

**Remark.** *It is important to note that we have not been able to verify whether Property (P) holds for the subproblem associated to the CETSP, and, as such, we cannot categorically state that GBD is not applicable to this problem. However, the invalidity of the cuts proposed by Gao et al. [8] highlights the challenges associated with applying GBD to this context, and suggests that further research is needed to explore the applicability of GBD to the CETSP and related problems.*

## 7 Computational experiments

In this section, we conduct computational experiments to test the different approaches presented in this work. In particular, we focus our analysis on the exact SOCP models (i.e., ABF, SBF) and the four decompositions (i.e., ABF-D, ABF-A-D, SBF-D, SBF-A-D) on a wide variety of instances. Then, we further analyze the quality of the approximated MILP (i.e., ABF-A-D, SBF-A-D), which are crucial for the good performance of our best-performing techniques.

We generate several CETSP instances, where $n \in \{10, 15, 20\}$ is the number of points, $r \in \{0.25, 0.5, 1\}$ the mean radii, and $\sigma \in \{0, 0.2, 0.5\}$ the variation among radii. The coordinates of the $n$ center points of the neighborhoods are chosen randomly from a uniform distribution on a square with side 10. Then, for each point, a radius is sampled uniformly from $[(1-\sigma)r, (1+\sigma)r]$. For each parameter combination, we generate 5 instances, yielding a total of 135 distinct instances. This instance-generation procedure is based on that of Behdani and Smith [2], adding the variability parameter $\sigma$ to generate more diverse instances. Approximations were implemented with $\nu = 3$, which yields an approximation factor of $\epsilon(\nu) < 0.02$. Before solving each instance, we consider a pre-processing step, where, if there is a pair $i, j \in N$ such that the neighborhood of $i$ is wholly contained in the neighborhood of $j$, we eliminate node $j$

from the problem, since it is trivially visited by any tour that visits the neighborhood of $i$.

We implement all procedures in Python 3.10, using Gurobi 12.0.2 as the solver, with a ten-minute time limit. All experiments are conducted on a PC equipped with an Intel Core i5-11300H 3.10GHz processor and 16GB of memory, running Windows 11. In the case of SBF-A-D, we report results using cuts (16a) and (16b), as they yielded slightly better results compared to using (17). A complete comparison of the two cut types is presented in Appendix A.

## 7.1 General results for exact formulations

We now review the results of the six exact procedures across 135 problem instances. Figure 4 presents a comparison of the performance of the six procedures as a function of execution time and the optimality gap. On the X-axis, values to the left of the vertical line represent time, while values to the right correspond to the optimality gap. The Y-axis indicates the number of instances solved. Each colored line represents the performance of a different formulation. The first vertical dotted line marks the 600 seconds time limit; thus, the left part of the plot shows the number of instances solved to optimality and the time needed to do so, while the right part of the plot presents information about the gap left on instances not solved to optimality.

Arc-based procedures (i.e., ABF, ABF-D, and ABF-A-D) are plotted in red, while sequence-based formulations (i.e., SBF, SBF-D, and SBF-A-D) are in blue. In both cases, the lighter shade corresponds to the base formulation, the medium shade to the decomposition, and the darker shade to the decomposition the MILP approximations of the SOC constraints.

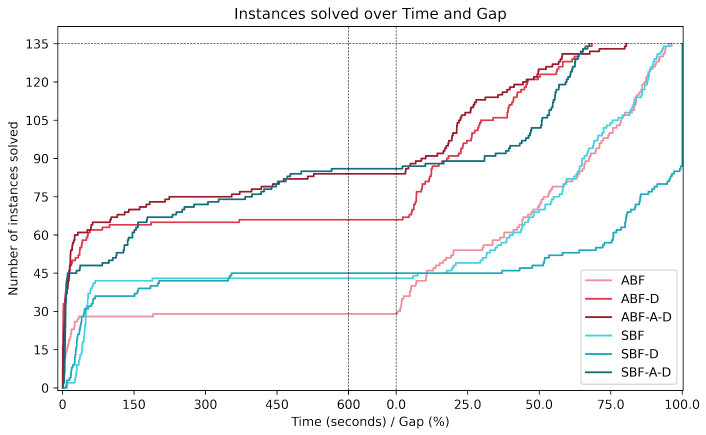

**Fig. 4**: Performance Profile Plot of exact formulations on all instances

We observe that the procedures that solve the largest number of instances to optimality are the decompositions based on the MILP approximations (i.e., SBF-A-D and ABF-A-D), with the first one solving slightly more instances. These are followed by ABF-D, and there is a significant difference separating these three formulations from the others. We can also observe that, besides SBF-A-D, all formulations find most of their optimal solutions in a fairly quick manner, and after about 300 seconds, the number of optimal solutions stagnates.

Table 3 supports these results, where column "Procedure" states the approach used, "#Opt" the number of optimal solutions found within the time limit, "TimeOpt" the average time on instances solved to optimality, "Gap" the average gap across all instances, and "Gap (NoOpt)" the average gap without considering instances solved to optimality. Lastly, the "#Cuts" column shows the average number of cuts generated.

| Procedure | #Opt | TimeOpt | Gap | Gap (NoOpt) | #Cuts |
|---|---|---|---|---|---|
| ABF | 29 | 17.42 | 0.43 | 0.54 | - |
| ABF-D | 66 | 21.87 | 0.16 | 0.30 | 12311.35 |
| ABF-A-D | 84 | 74.18 | 0.12 | 0.33 | 109.27 |
| SBF | 43 | 46.29 | 0.43 | 0.63 | - |
| SBF-D | 45 | 73.06 | 0.60 | 0.89 | 2232.52 |
| SBF-A-D | 86 | 119.36 | 0.18 | 0.51 | 599.11 |

**Table 3**: Summary of results for all instances

We observe that, overall, ABF-A-D has the best performance, with only two less instances solved to optimality than SBF-A-D, but with a considerably smaller average gap, both overall and on instances not solved to optimality. With regards to the second and third formulations with best performance, SBF-A-D and ABF-D, we note that SBF-A-D solves 20 more instances to optimality, but with substantially larger average gaps, and ABF-D has smaller average runtime on optimal instances. Additionally, we note that, while ABF and SBF cannot be categorically ranked, with SBF achieving a greater number of optimalities and ABF smaller gaps on non-optimal instances, ABF-D seems to be a considerably better formulation than its sequence-based counterpart, obtaining smaller gaps and more optimalities in less average time. In fact, SBF-D appears to be a fairly weak decomposition, obtaining only two more optimalities than SBF, but with quite larger gaps on instances not solved to optimality.

It is also interesting to note that both non approximation-based arc formulations have average running times on optimal instances under 22 seconds, suggesting that, at least with the time limit imposed, these formulations either find an optimal solution quickly or fail to find an optimal solution altogether.

The superior performance of the SBF-A-D and ABF-A-D decompositions can be explained by the higher quality of the distance estimation used in the master problem. These formulations leverage the quality of the approximation of the conic constraints to obtain fairly accurate cost estimations, thus lessening the need for refinement via cut generation. On the other hand, the cost estimations used in ABF-D are fairly lax and have the potential to severely underestimate actual costs, leading to the need for a large number of cuts to obtain good solutions due to the enumerative nature of the cuts. This is also a possible reason for the poor performance of SBF-D, since it has no proper cost estimation in the master problem, thus obtaining all information about the tour length from the cuts generated in the subproblem. Indeed, ABF-A-D and SBF-A-D generate two orders of magnitude fewer cuts than ABF-D, which suggests that the quality of the master problem cost estimation is a key factor in the performance of these decompositions, as it can significantly reduce the need for cut generation.

Lastly, it is worth noting that the three best-performing approaches all use enumerative cuts, suggesting that there is still room for improvement in this regard, as stronger cuts could yield considerably better results.

## 7.2 Results by instance type for exact formulations

To provide an in-depth analysis of performance across different instance types, we present plots and tables similar to those in Section 7.1, grouping instances by the number of points $n$ and the parameter $r$ (i.e., mean neighborhood radii) used to generate them. We do not report the analysis grouped by $\sigma$ value (i.e., variation among radii), since different values of this parameter did not lead to significant performance variations, at least within the tested range.

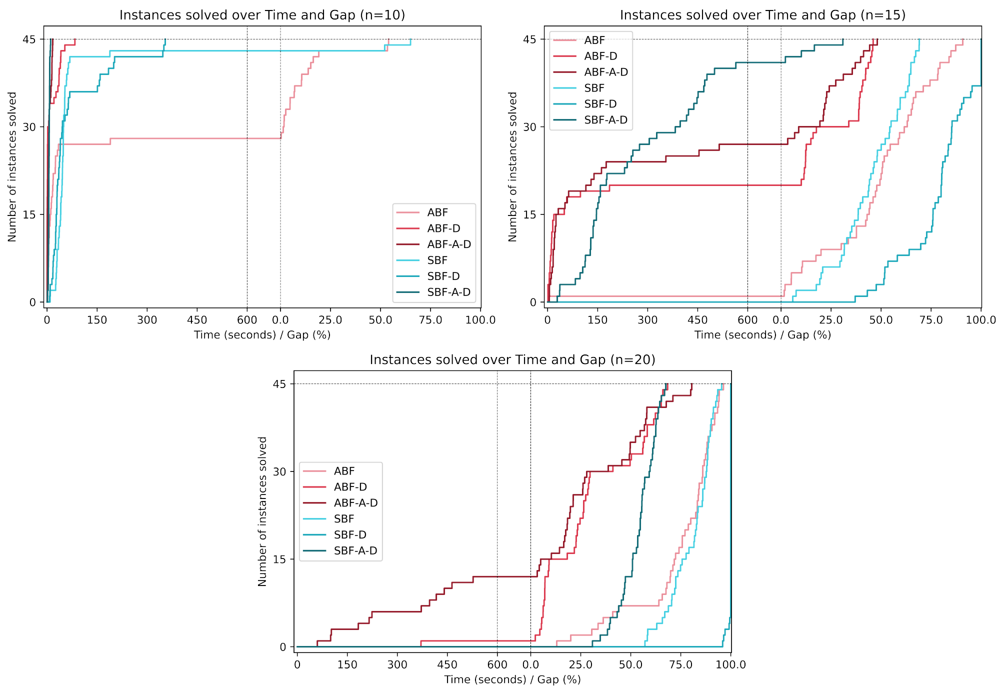

**Fig. 5**: Performance Profile Plot of exact formulations grouped by amount of points

| $n$ | Formulation | #Opt | TimeOpt | Gap | Gap (NoOpt) | #Cuts |
|---|---|---|---|---|---|---|
| 10 | ABF | 28 | 17.88 | 0.05 | 0.13 | - |
| 10 | ABF-D | 45 | 11.00 | 0.00 | - | 1433.38 |
| 10 | ABF-A-D | 45 | 5.44 | 0.00 | - | 51.78 |
| 10 | SBF | 43 | 46.29 | 0.03 | 0.59 | - |
| 10 | SBF-D | 45 | 73.06 | 0.00 | - | 1656.38 |
| 10 | SBF-A-D | 45 | 5.79 | 0.00 | - | 201.96 |
| 15 | ABF | 1 | 4.35 | 0.49 | 0.50 | - |
| 15 | ABF-D | 20 | 28.89 | 0.17 | 0.30 | 15485.91 |
| 15 | ABF-A-D | 27 | 92.07 | 0.11 | 0.26 | 136.18 |
| 15 | SBF | 0 | - | 0.45 | 0.45 | - |
| 15 | SBF-D | 0 | - | 0.79 | 0.79 | 3030.91 |
| 15 | SBF-A-D | 41 | 244.02 | 0.01 | 0.15 | 1301.64 |
| 20 | ABF | 0 | - | 0.74 | 0.74 | - |
| 20 | ABF-D | 1 | 370.72 | 0.30 | 0.31 | 20014.76 |
| 20 | ABF-A-D | 12 | 291.72 | 0.26 | 0.36 | 139.87 |
| 20 | SBF | 0 | - | 0.81 | 0.81 | - |
| 20 | SBF-D | 0 | - | 1.00 | 1.00 | 2010.27 |
| 20 | SBF-A-D | 0 | - | 0.54 | 0.54 | 293.73 |

**Table 4**: Summary of results for instances grouped by amount of points.

***Number of points $n$ analysis.***

Figure 5 and Table 4 present the results grouped by the number of visited points per instance. We observe that, on instances with 10 points, ABF-A-D has the best performance, followed closely by SBF-A-D and ABF-D, with all three of these formulations solving the whole set of instances to optimality in under 12 seconds. We also observe that SBF has a better overall performance than ABF on these instances, only failing to achieve optimality on two of them, although ABF finds optimal solutions faster when it does find them. Lastly, it is worth noting that SBF-D solves all instances of this set to optimality.

In instances with 15 points, the most notable feature is the performance of SBF-A-D, which solves all but 4 of the 45 instances to optimality and achieves very small gaps on the remaining 4, a substantially better performance than all other formulations. Once again, the second-best performer is ABF-A-D, followed closely by ABF-D. In contrast to the case with 10 points, here SBF-D fails to solve any instance to optimality, and, in fact, is categorically outperformed by all other formulations. Finally, ABF has similar performance to its sequence-based counterpart, solving one instance to optimality but with larger average gaps.

Lastly, on instances with 20 points, we observe a considerable drop in performance across all formulations, with only ABF-D and ABF-A-D being able to solve any instance to optimality, with the latter being the best performer, solving 12 of the 45 instances. We also note that SBF-A-D, which had been the best performer on instances with 15 points, fails to solve any instance to optimality here, and has a considerably larger average gap than ABF-A-D. Finally, we observe that ABF and SBF have similar performances, with ABF having a slightly smaller average gap, and SBF-D not only fails to solve any instance to optimality, but also has a gap of 100% on all instances.

In conclusion, these results imply a trade-off between overall performance and scalability, with SBF-A-D being the strongest formulation on small and medium instances, and both arc-based decompositions, ABF-D and ABF-A-D, being the only formulations to be able to solve instances with $n = 20$ and providing the lowest gaps for this set of instances. The following analysis reveals that the overall performance advantage of the arc-based formulations on large instances is driven by a specific subset (namely, those with small radii) on which arc-based formulations perform exceptionally well.

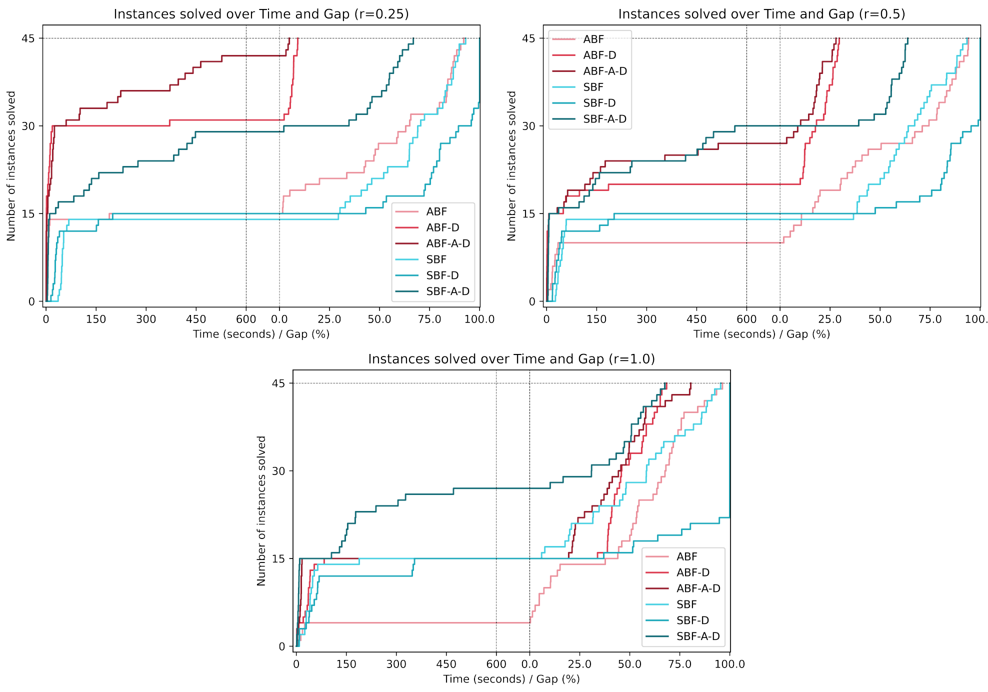

**Fig. 6**: Performance Profile Plot of exact formulations grouped by radius values.

***Mean Radii $r$ analysis.***

Figure 6 and Table 5 present the results grouped by the value of $r$ used to generate the instances. In instances with small radii, (i.e., $r = 0.25$) with any value of $\sigma$, we observe that the three formulations that showed the best performance overall, ABF-A-D, SBF-A-D, and ABF-D, are also the best performers. However, in this set, ABF-D and ABF-A-D perform considerably better than SBF-A-D, with ABF-A-D finding all but 3 optimal solutions, and ABF-D finding 2 more optimal solutions than SBF-A-D, but in considerably lesser times, and with smaller gaps on non-optimal instances.

| $r$ | Formulation | #Opt | TimeOpt | Gap | Gap (NoOpt) | #Cuts |
|---|---|---|---|---|---|---|
| 0.25 | ABF | 15 | 17.12 | 0.38 | 0.58 | - |
| 0.25 | ABF-D | 31 | 16.48 | 0.02 | 0.07 | 7572.98 |
| 0.25 | ABF-A-D | 42 | 89.35 | 0.00 | 0.04 | 77.29 |
| 0.25 | SBF | 14 | 49.58 | 0.46 | 0.67 | - |
| 0.25 | SBF-D | 15 | 55.75 | 0.58 | 0.86 | 2096.04 |
| 0.25 | SBF-A-D | 29 | 116.42 | 0.18 | 0.50 | 397.78 |
| 0.50 | ABF | 10 | 18.49 | 0.42 | 0.54 | - |
| 0.50 | ABF-D | 20 | 23.70 | 0.11 | 0.20 | 13114.40 |
| 0.50 | ABF-A-D | 27 | 85.53 | 0.07 | 0.18 | 85.60 |
| 0.50 | SBF | 14 | 42.33 | 0.45 | 0.65 | - |
| 0.50 | SBF-D | 15 | 60.75 | 0.60 | 0.90 | 2100.73 |
| 0.50 | SBF-A-D | 30 | 142.70 | 0.19 | 0.57 | 417.24 |
| 1.00 | ABF | 4 | 15.84 | 0.47 | 0.51 | - |
| 1.00 | ABF-D | 15 | 30.58 | 0.33 | 0.50 | 16246.67 |
| 1.00 | ABF-A-D | 15 | 11.28 | 0.29 | 0.44 | 164.93 |
| 1.00 | SBF | 15 | 46.90 | 0.37 | 0.56 | - |
| 1.00 | SBF-D | 15 | 102.68 | 0.61 | 0.92 | 2500.78 |
| 1.00 | SBF-A-D | 27 | 96.59 | 0.19 | 0.47 | 982.31 |

**Table 5**: Summary of results for instances grouped by radius values.

Additionally, we note that ABF also dominates its sequence-based counterpart SBF, which has a performance relatively similar to that of its decomposition SBF-D.

On instances with medium radii (i.e.,$r = 0.5$), with any value of $\sigma$, we observe a different behavior. In particular, we note that SBF-A-D is the formulation that solves the most instances to optimality, but its overall performance is arguably worse than that of ABF-A-D, which solves only 3 fewer instances, but has significantly smaller gaps on non-optimal instances. ABF-D shows a performance slightly worse than that of ABF-A-D, with similar average gaps, but solving 7 fewer instances to optimality. We also note that SBF seems to perform better than ABF at obtaining optimal solutions, achieving 4 more than ABF, but its average gap is larger.

Finally, on instances generated with $r = 1$, we observe that SBF-A-D presents an exceptional performance, finding optimal solutions for 27 out of 45 instances, while achieving gaps similar to those of ABF-A-D in the non-optimal cases. In contrast, no other formulation finds more than 15 optimal solutions. Once again, ABF-A-D shows the second-best performance on this set. We also observe that, contrary to what happened in the small radii regime, SBF has a substantially better performance than ABF.

In general, we note that the problem becomes more difficult as the radii increase. This is particularly evident on all three arc-based formulations, which suffer a significant performance drop as the radii increase. In contrast, the sequence-based formulations maintain a more consistent performance across different radii, but still show slightly worse results overall. We theorize that this behavior is because, for smaller radii, the problem more closely resembles the classical TSP, where the formulations analog to ABF are known to perform considerably better than those analog to SBF. However, as the radii increase, the problem diverges from the classical TSP, and the advantages of arc-based formulations are less pronounced.

## 7.3 Results for approximate formulations

Since the decompositions based on the approximation of the second-order cone (SBF-A-D and ABF-A-D) are the ones with the best overall performance, it is of interest to analyze the quality of the approximations by themselves, that is, SBF-A and ABF-A. To do this, Figure 7 presents scatter plots comparing the runtime and the optimality gap of these two approximate formulations, where each point corresponds to a different instance. The X-axis of the first plot represents the runtime of ABF-A, while the Y-axis represents the runtime of SBF-A. The second plot has the same axes, but now they represent the optimality gap for each formulation, i.e., the difference between the lower bound obtained from the approximation and the lower bound from solving the exact SOCP formulation with the visit order fixed by the solution of the approximation. The diagonal line is the identity; thus, points above this line represent instances where the values associated with SBF-A are greater than those associated with ABF-A, while points below the line represent instances where the opposite is true. Different marker shapes and colors are used according to the values of $n$ and $r$ corresponding to each instance. Additionally, Table 6 summarizes the results of both formulations, with columns for the formulation used, the average runtime, the average gap, and the number of instances where the approximation was solved to optimality.

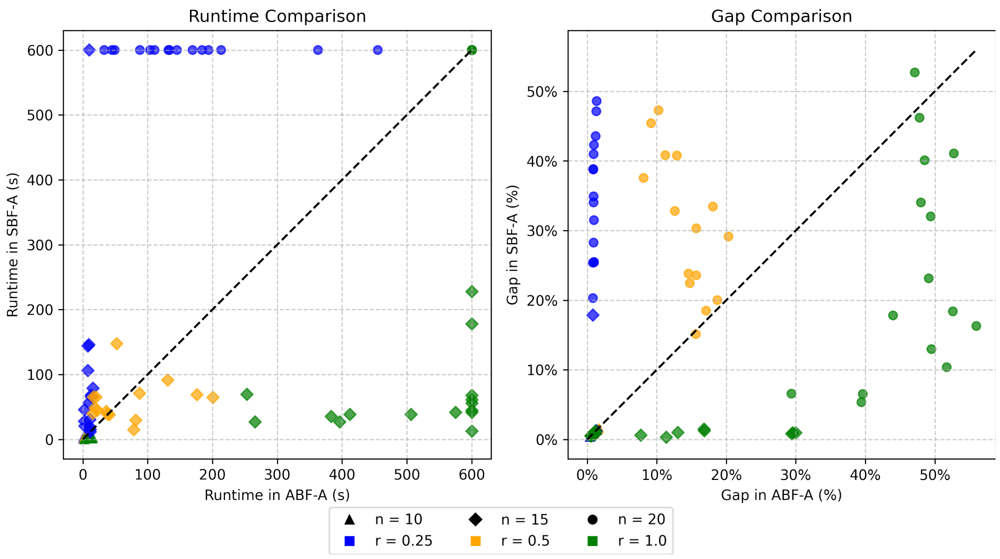

**Fig. 7**: Runtime and gap comparisons for approximate MILP formulations

With regard to the runtimes, we observe that SBF-A either solves the instance to optimality in under 200 seconds or fails on time. On the other hand, ABF-A also solves most instances quickly, but has more variability in this regard. Despite some instances reaching the time limit, both formulations solve most instances in less than 100 seconds, proving the approximations to be tractable in practice.

| Formulation | Avg. Runtime (s) | Avg. Gap (%) | # Solved |
|---|---|---|---|
| ABF-A | 217.51 | 8.61 | 97 |
| SBF-A | 224.97 | 10.74 | 89 |

**Table 6**: Results summary for approximate formulations

In terms of optimality gap, we observe that on most instances the gap obtained is the same for both approximations, with values under 2.5%. In fact, instances where the gaps obtained by the approximations differ are precisely those where one of them fails to find an optimal solution. Lastly, we note that, for both approximations, the harder instances are those with $n = 20$, with ABF-A having a considerably better performance on instances with small and medium radii out of this set, while the exact opposite happens on instances with large radii.

These results suggest that the approximations are fairly strong, offering high-quality solutions without incurring excessive computational costs, with the arc-based approximation being slightly superior overall to the sequence-based one, which further supports the idea that ABF-A-D and SBF-A-D are the best performing exact formulations due to the quality of the approximations they are based on leading to accurate master problem cost estimations, thus reducing the need for cut generation.

Additionally, the results obtained explain the differences in performance between ABF-A-D and SBF-A-D when grouping by value of $r$. The underlying approximations for these decompositions exhibit similar trends: the arc-based approximation is stronger on small and medium radii, while the sequence-based one has greater performance on larger radii. These performance patterns explain the overall outcomes for the full decompositions, as a stronger performance of the approximation on an instance should lead to a shorter runtime for the corresponding decomposition. Lastly, the observation that SBF-A reaches the time limit on more $n = 20$ instances than ABF-A partly explains the final performance, as this relative weakness of the sequence-based approximation is clearly mirrored in the full decomposition SBF-A-D.

## 8 Conclusions

This work addresses the Close-Enough Traveling Salesman Problem (CETSP). We present two exact formulations for this problem based on second-order cone programming, along with approximated mixed-integer linear formulations derived from polyhedral approximations of the second-order cone. Additionally, we propose decomposition schemes for both exact and approximate formulations, utilizing cut generation to iteratively refine the master problem.

Our computational experiments demonstrate the effectiveness of the proposed formulations and decomposition schemes, with the decompositions based on the approximated formulations showing the best overall performance. We also identify different instance regimes where each formulation excels, providing insights into their strengths and weaknesses. Furthermore, we critically analyze the decomposition scheme proposed by Gao et al. [8], highlighting issues with the validity of the cuts used and discussing the implications for convergence to optimality.

Future research could explore the development of stronger cuts for the decomposition schemes, especially considering the enumerative nature of the cuts used in the best-performing decompositions. Additionally, the formulations and decomposition schemes presented could be extended to address related problems, such as the Prize-Collecting TSP with neighborhoods or the Vehicle Routing Problem with neighborhoods. Finally, the decomposition approach based on the approximation of the second-order cone could be adapted to solve other optimization problems involving conic constraints and binary variables, potentially leading to new insights and solution methods in this area.

## Appendix A Comparison of cuts for SBF-A-D

In figure A1, we present a performance profile plot comparing the performance of SBF-A-D when using cuts (16a) and (16b), which we note LL cuts, against the performance when using (17), which we denote Dual cuts, and the performance when adding both types of cuts. Detailed results are presented in Table A1.

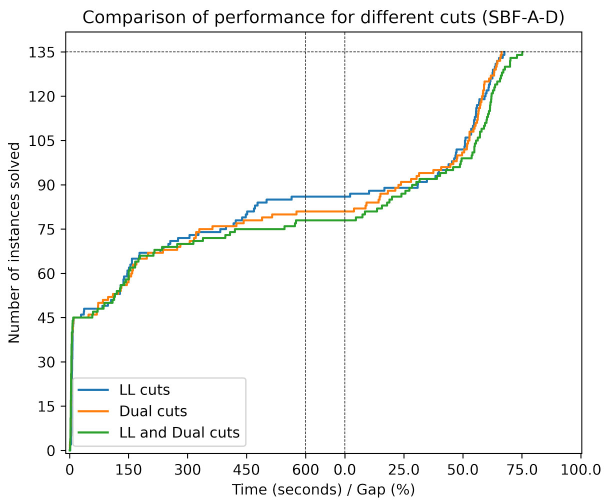

**Fig. A1**: Performance Profile Plot of SBF-A-D with different cut types

| Cut type | #Opt | TimeOpt | Gap | Gap (NoOpt) | #Cuts |
|---|---|---|---|---|---|
| LL cuts | 86 | 119.36 | 0.18 | 0.51 | 599.11 |
| Dual cuts | 81 | 101.49 | 0.19 | 0.47 | 198.27 |
| LL and Dual cuts | 78 | 93.99 | 0.21 | 0.49 | 480.24 |

**Table A1**: Summary of results for SBF-A-D with different cut types

We observe that there is no performance gain from adding both types of cuts, while the performances when using only one type of cut are fairly similar, with LL cuts having a slight edge overall. It should also be noted that the number of cuts generated when using Dual cuts is considerably smaller than when using LL cuts, which suggests that these cuts are stronger in nature, but the overall performance indicates that the simplicity of LL cuts makes them more effective in practice.

## References

[1] Applegate, D.L., Bixby, R.E., Chvátal, V., Cook, W.J.: The Traveling Salesman Problem: A Computational Study. Princeton University Press, Princeton (2007). https://doi.org/10.1515/9781400841103

[2] Behdani, B., Smith, J.C.: An integer-programming-based approach to the close-enough traveling salesman problem. INFORMS Journal on Computing **26**(3), 415–432 (2014) https://doi.org/10.1287/ijoc.2013.0574

[3] Coutinho, W.P., Nascimento, R.Q.d., Pessoa, A.A., Subramanian, A.: A branch-and-bound algorithm for the close-enough traveling salesman problem. INFORMS Journal on Computing **28**(4), 752–765 (2016) https://doi.org/10.1287/ijoc.2016.0711

[4] Di Placido, A., Archetti, C., Cerrone, C.: A genetic algorithm for the close-enough traveling salesman problem with application to solar panels diagnostic reconnaissance. Computers & Operations Research **145**, 105831 (2022) https://doi.org/10.1016/j.cor.2022.105831

[5] Yuan, b., Orlowska, M., Sadiq, S.: On the optimal robot routing problem in wireless sensor networks. IEEE Trans. Knowl. Data Eng. **19**, 1252–1261 (2007) https://doi.org/10.1109/TKDE.2007.1062

[6] Di Placido, A., Archetti, C., Cerrone, C., Golden, B.: The generalized close enough traveling salesman problem. European Journal of Operational Research **310**(3), 974–991 (2023) https://doi.org/10.1016/j.ejor.2023.04.010

[7] Carrabs, F., Cerulli, R., D'Ambrosio, C., Murano, G.: Upper bound computation for the multiple close-enough traveling salesman problem, pp. 186–195 (2025). https://doi.org/10.5220/0013377900003893

[8] Gao, C., Wei, N., Walteros, J.L.: An exact approach for solving pickup-and-delivery traveling salesman problems with neighborhoods. Transportation Science **57**(6), 1560–1580 (2023) https://doi.org/10.1287/trsc.2022.0138

[9] Geoffrion, A.M.: Generalized Benders decomposition. Journal of optimization theory and applications **10**, 237–260 (1972)

[10] Mennel, W.K.: Heuristics for solving three routing problems: Close-enough traveling salesman problem, close-enough vehicle routing problem, sequence-dependent team orienteering problem. PhD thesis, The Robert H. Smith School of Business, University of Maryland (2009)

[11] Lei, Z., Hao, J.-K.: An effective memetic algorithm for the close-enough traveling salesman problem. Applied Soft Computing **153**, 111266 (2024) https://doi.org/10.1016/j.asoc.2024.111266

[12] Gentilini, I., Margot, F., Shimada, K.: The travelling salesman problem with neighbourhoods: MINLP solution. Optimization Methods and Software **28**(2), 364–378 (2013)

[13] Carrabs, F., Cerrone, C., Cerulli, R., Gaudioso, M.: A novel discretization scheme for the close enough traveling salesman problem. Computers & Operations Research **78**, 163–171 (2017) https://doi.org/10.1016/j.cor.2016.09.003

[14] Zhang, W., Sauppe, J., Jacobson, S.: Results for the close-enough traveling salesman problem with a branch-and-bound algorithm. Computational Optimization and Applications **85**, 1–39 (2023) https://doi.org/10.1007/s10589-023-00474-3

[15] Gutow, G., Choset, H.: Efficient second-order cone programming for the close enough traveling salesman problem

[16] Deckerová, J., Váňa, P., Faigl, J.: Combinatorial lower bounds for the generalized traveling salesman problem with neighborhoods. Expert Systems with Applications **258**, 125185 (2024) https://doi.org/10.1016/j.eswa.2024.125185

[17] Gavish, B., Graves, S.C.: The travelling salesman problem and related problems. (1978)

[18] Raturi, A., Tsubakitani, S.: Cutting Big M down to size. Interfaces **20**, 61–66 (1990) https://doi.org/10.1287/inte.20.5.61

[19] Ben-Tal, A., Nemirovski, A.: On polyhedral approximations of the second-order cone. Mathematics of Operations Research **26**(2), 193–205 (2001). Accessed 2025-02-19

[20] Laporte, G., Louveaux, F.V.: The integer L-shaped method for stochastic integer programs with complete recourse. Operations Research Letters **13**(3), 133–142 (1993) https://doi.org/10.1016/0167-6377(93)90002-X

[21] Borwein, J.J., Lewis, A.: Convex Analysis and Nonlinear Optimization: Theory and Examples, (2005). https://doi.org/10.1007/978-0-387-31256-9

[22] Benders, J.F.: Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik **4**(1), 238–252 (1962) https://doi.org/10.1007/BF01386316 . Cited by: 2963

[23] Costa, A.M.: A survey on Benders decomposition applied to fixed-charge network design problems. Computers & Operations Research **32**(6), 1429–1450 (2005) https://doi.org/10.1016/j.cor.2003.11.012

[24] Bagajewicz, M.J., Manousiouthakis, V.: On the Generalized Benders Decomposition. Computers & Chemical Engineering **15**(10), 691–700 (1991) https://doi.org/10.1016/0098-1354(91)85015-M