A tabu search with geometry-based sparsification methods for angular traveling salesman problems

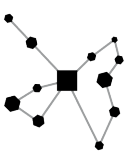
Working Paper DPO-2022-01 (version 0, 13.07.2022)

Rossana Cavagnini, Michael Schneider, Alina Theiß
{cavagnini|schneider|theiss}@dpo.rwth-aachen.de
Deutsche Post Chair – Optimization of Distribution Networks
RWTH Aachen University, Germany

Abstract

The angular-metric traveling salesman problem (AngleTSP) aims to find a tour visiting a given set of vertices in the Euclidean plane exactly once while minimizing the cost given by the sum of all turning angles. If the cost is obtained by combining the sum of all turning angles and the traveled distance, the problem is called angular-distance-metric traveling salesman problem (AngleDistanceTSP). In this work, we study the symmetric variants of these problems. Because both the AngleTSP and AngleDistanceTSP are NP-hard, multiple heuristic approaches have been proposed in the literature. Nevertheless, a good tradeoff between solution quality and runtime is hard to find. We propose a granular tabu search (GTS) that considers the geometric features of the two problems in the design of starting solutions and sparsification methods. We further enrich the GTS with components that guarantee both intensification and diversification during the search. The computational results on benchmark instances from the literature show that (i) for the AngleTSP, our GTS lies on the Pareto frontier of the best performing-heuristics, and (ii) for the AngleDistanceTSP, our GTS provides the best solution quality across all existing heuristics in competitive runtimes. In addition, new best-known solutions are found for most benchmark instances for which an optimal solution is not available.

Keywords: *Angular TSPs, AngleTSP, AngleDistanceTSP, granular tabu search, sparsification methods*



Deutsche Post
Chair - Optimization of
Distribution Networks

RWTH AACHEN
UNIVERSITY

1 Introduction

The angular-metric traveling salesman problem (AngleTSP) aims to find a Hamiltonian cycle visiting a given set of vertices in the Euclidean plane while minimizing the cost given by the sum of all turning angles. If the cost is obtained by combining the sum of all turning angles and the traveled distance, the problem is called angular-distance-metric traveling salesman problem (AngleDistanceTSP). The two problems have applications in, e.g., robotics and transportation. In robotics, keeping the movements of a robot as straight as possible is key to avoiding energy-consuming changes in driving direction. In transportation, straight movements make heavy vehicles more controllable.

In this work, we consider the symmetric variants of these problems, which are formally defined as follows. Let $G = (V,A)$ be a complete directed graph with vertex set $V = \{1,2,\ldots,n\}$ in the Euclidean plane and arc set $A = \{(i,j) : i,j \in V, i \neq j\}$. A non-negative distance d_{ij} is associated with each arc and a non-negative turning angle α_{ijk} with each vertex triplet $(i,j,k), i,j,k \in V$. The cost for each triplet (i,j,k) is defined as $c_{ijk} = \rho_1\alpha_{ijk} + \rho_2(d_{ij} + d_{jk})/2$, where ρ_1 and ρ_2 are the weights associated with the turning angles and the distances, respectively. For the AngleTSP, ρ_2 is equal to 0. A solution for the AngleTSP or AngleDistanceTSP is denoted by S, and it is a tour visiting each vertex in V exactly once. Its cost is denoted by $C(S)$. A solution is optimal if it minimizes the cost of the tour.

Because both problems are NP-hard and exact solution methods can only solve small instances in reasonable runtimes (see Fischer et al. 2014), Staněk et al. (2019) proposed multiple heuristics. The best-performing methods are either simple construction heuristics or matheuristics: the former are fast but provide unsatisfying solution quality, the latter achieve superior solution quality but are expensive in terms of runtime. There is evidence of a gap in the literature concerning the availability of fast but effective heuristics for both problem variants.

This paper contributes to filling this gap by presenting a granular tabu search (GTS) framework, called *GTS-angular*, to address both problem variants. To obtain decent starting solutions for GTS-angular, we exploit geometric properties of good solutions. For the AngleTSP, we present a faster version of the convex hull construction heuristic proposed by Staněk et al. (2019). For the AngleDistanceTSP, we develop a new construction method that makes use of the fact that good solutions typically consist of a tour in which arcs do not intersect.

To speed up the search, our tabu search adopts the granular search principle introduced by Toth and Vigo (2003). This principle relies on sparsification methods which are used to restrict the size of the neighborhoods to explore. GTS frameworks provide high-quality solutions within reduced runtimes for several routing-related problems like the capacitated vehicle routing problem (CVRP, Toth and Vigo 2003), the vehicle routing problem (VRP) with time windows (Schneider et al. 2017), the multi-depot VRP (Escobar et al. 2014), the pickup and delivery problem with time windows and electric vehicles (Goeke 2019), the dial-a-ride problem (Kirchler and Wolfler Calvo 2013), and the capacitated location routing problem (CLRP, Prins et al. 2007, Schneider and Löffler 2019).

In this work, we strengthen the sparsification method originally proposed by Staněk et al. (2019) of using a lens to limit the size of the neighborhood to search. Our contribution is to design two problem-specific sparsification methods which are applied to both the AngleTSP and AngleDistanceTSP and which give rise to two algorithmic variants of GTS-angular.

We run numerical studies on benchmark instances from the literature to test the performance of the two GTS-angular variants differing with respect to the sparsification method. Compared with the best-performing heuristics developed by Staněk et al. (2019), GTS-angular improves either the solution quality or the runtimes, sometimes both, and finds new best-known solutions for many benchmark instances.

The remainder of the paper is organized as follows. In Section 2, we review the literature related to the AngleTSP and the AngleDistanceTSP. Section 3 describes GTS-angular in detail. We discuss the experimental setting and the computational results in Section 4. Finally, Section 5 presents our conclusion.

2 Literature review

The AngleTSP has been introduced by Aggarwal et al. (2000), who show that this problem is NP-hard. Savla et al. (2008) are the first authors to study the AngleDistanceTSP, which is later formalized by Medeiros and Urrutia (2010) as an approximation of the TSP in which the turn radius of the car is limited.

In both problem variants, the cost arises for the successive use of two arcs. Hence, most research has treated these problems as generalizations of the quadratic TSP (QTSP). Polyhedral properties for the symmetric QTSP are studied in Fischer and Helmberg (2013), who derive classes of strengthened subtour elimination constraints. Fischer et al. (2014) study the symmetric QTSP and propose both exact and heuristic approaches to solve it. The exact approaches include: a transformation of the problem to the TSP, then solved by standard methods, a branch-and-bound algorithm, and a branch-and-cut algorithm. These methods solve instances with up to 40 vertices in reasonable runtimes. The heuristic approaches are based on modifications of heuristics commonly applied to the TSP. While these methods are faster than the exact ones, they show significant optimality gaps even for relatively small instances with up to 50 vertices. Fischer et al. (2015) investigate the performance of three exact methods for solving QTSP instances with bioinformatic applications.

Apart from Aggarwal et al. (2000), who exclusively studies the AngleTSP, the only paper in the literature focusing exclusively on the AngleTSP and AngleDistanceTSP is the one of Staněk et al. (2019). The authors propose a large number of heuristic approaches that can be classified into the following categories: construction heuristics, geometric heuristics (applied only to the AngleTSP), and linear programming-based approaches. The authors extend these methods by applying improvement algorithms, such as 2-opt and 3-opt heuristics, a destroy-and-repair matheuristic, and a simulated annealing metaheuristic. Their fastest methods are the simple construction heuristics, and the methods returning the best solution quality are the linear programming-based approaches improved by the destroy-and-repair matheuristic. A good trade-off between solution quality and runtime is difficult to find. Moreover, the authors highlight that the quite notable optimality gaps make the development of metaheuristics a promising research topic.

3 Granular tabu search for the angular metric traveling salesman problem

In this section, we describe the general idea of tabu search (TS) and we introduce the GTS-angular framework. Section 3.1 provides details on how the starting solution is obtained, Section 3.2 describes the used neighborhoods, Section 3.3 introduces the sparsification methods, and Section 3.4 presents the continuous diversification strategy.

The TS framework was introduced by Glover (1989). In its basic form, it includes a local search that iteratively explores neighboring solutions. Two main features of TS are (i) the presence of a memory, which contains information on which moves are tabu to avoid cycling, and (ii) the application of the best non-tabu move in each iteration—which may be deteriorating—to escape from local optima.

Figure 1 provides a pseudocode overview of GTS-angular and its main components are detailed in the remainder of this section. After obtaining a starting solution according to the construction heuristic described in Section 3.1 (line 1) and initializing the current solution S^{curr} and the overall best-found solution S^* with it, the TS component is initialized (line 2) and executed (lines 3–9). The tabu criterion forbids to reinsert an arc that has been removed for a tenure of τ iterations. As aspiration criterion, we always permit a move that is tabu but improves on the overall best-found solution S^*.

In each iteration, GTS-angular evaluates neighboring solutions that are obtained by applying a move defined by a so-called generator arc and a neighborhood operator. The neighborhood operators are designed in such a way that the generator arc is always an arc inserted in the solution. Our algorithm first traverses the generator arc set A_g (line 5). To speed up the search, the dimension of A_g is reduced to the arcs selected by applying the sparsification methods described in Section 3.3. For each generator arc, we loop over the relocate, exchange, and 2-opt operators contained in the set $\mathscr{O}$ (line 6) and presented in Section 3.2, and the resulting move is evaluated (line 7). Moves deteriorating the cost are evaluated according to an extended objective function that considers the number of times the inserted arcs were included in previously carried-out moves (Section 3.4). This continuous diversification mechanism (lines 8–10) guides the search to new areas of the solution space by favoring solutions with arcs that have rarely been included in previous iterations. In each iteration of the TS, we carry out the best non-tabu move (lines 11–17). Note that, we have also tested the first-improvement variant of GTS-angular during our computational experiments. However, the gained runtime advantage was not high enough to counterbalance the experienced loss in solution quality. Consequently, we only focus on the best-improvement variant of GTS-angular. If the move is tabu and does not satisfy the aspiration criterion (lines 11–13), TS omits the cost check and the eventual update of the current solution (lines 14–17). Finally, the overall best solution, the tabu list, and the generator arc set are updated (line 20). GTS-angular terminates after η iterations without improvement.

3.1 Construction heuristics

To generate an initial solution, we use separate algorithms for the AngleTSP (Section 3.1.1) and the AngleDistanceTSP (Section 3.1.2).

3.1.1 Initial solution for the AngleTSP

For the AngleTSP, the literature has shown that optimal tours have a snail shell shape (see, e.g., Aichholzer et al. 2017, Staněk et al. 2019). Therefore, we use a construction heuristic based on convex hulls similar to the one described in Staněk et al. (2019). Our construction heuristic works as follows. We first determine the convex hull considering the complete vertex set V. While Staněk et al. (2019) does not specify how the convex hulls are computed, we use a modified version of Graham's scan algorithm (Graham 1972). This modified version is described by Cormen et al. (2009): instead of computing the actual polar angles via trigonometric functions and sorting the vertices accordingly in increasing order, vertices are sorted by only comparing the polar angles via cross products. Given are a reference vertex v_0, which is the vertex with the

Pseudocode of GTS-angular

```
S^curr ← constructionHeuristic()
initialize S* ← S^curr, tabuList, and generator arc set A_g
while termination criterion not satisfied do
    C(S) ← ∞
    for (i, j) ∈ A_g do
        for o ∈ Ō do
            S' ← move((i,j), o, S^curr)
            if C(S') ≥ C(S^curr) then
                C(S') ← continuousDiversification()
            end
            if S' ∈ tabuList and notAspiration then
                continue
            end
            if C(S') < C(S) then
                S^curr ← S'
                C(S) ← C(S')
            end
        end
    end
    update S*, tabuList, A_g
end
```

Figure 1: Pseudocode of GTS-angular.

minimum y-coordinate and a pair of vertices v_1 and v_2. To decide whether the polar angle of v_1 with respect to v_0 is larger or smaller than the polar angle of v_2 with respect to v_0, a single cross product can be computed without computing the actual angle. If the sign of the cross product is negative, then a left turn must be taken to move from v_0 to v_1 to v_2, i.e., the polar angle is wider for v_2 than for v_1. If the sign is positive, a right turn must be taken, i.e., the polar angle is smaller for v_2 than for v_1. The special case of a cross product equal to zero represents collinearity of vertices v_0, v_1 and v_2. In this case, the vertex farther away from v_0 is chosen first.

Once the outermost convex hull is found, we remove the vertices defining this convex hull from V, and we compute the convex hull on the remaining vertices. We repeat the procedure until at most one vertex is left. Finally, these convex hulls (and the final vertex, if any) are connected to form a tour in the cheapest insertion fashion from the innermost to the outermost convex hull. We merge the convex hulls in this order because Staněk et al. (2019) observe that the cost of the solution increases more when the innermost convex hulls are merged. Moreover, we do not adopt any look-ahead strategy that makes a merging decision by considering the future ones. Consequently, by merging the convex hulls in this order, we provide more flexibility of making good merging decisions for convex hulls that contribute to the solution cost the most. Specifically, given a pair of convex hulls, the inner one is inserted in the outer one at the position causing the minimal cost increase. If the innermost "convex hull" consists of just two vertices, they are inserted at once. Our insertion procedure makes our construction heuristic faster than the procedure described in Staněk et al. (2019). In their approach, every time two convex hulls must be merged, the authors check, for all pairs of convex hulls and all edge pairs, the connection returning the smallest sum of objective function values over all subtours. Figure 2 shows an example of constructing an initial solution for the AngleTSP according to our approach. In step 1, we obtain three convex hulls corresponding to three subtours. In step 2, we connect the two innermost convex hulls in cheapest insertion fashion, and two subtours are left. Finally, in step 3, the innermost subtour is connected to the outermost convex hull. Because this last step results in a tour visiting all vertices in V, the algorithm terminates.

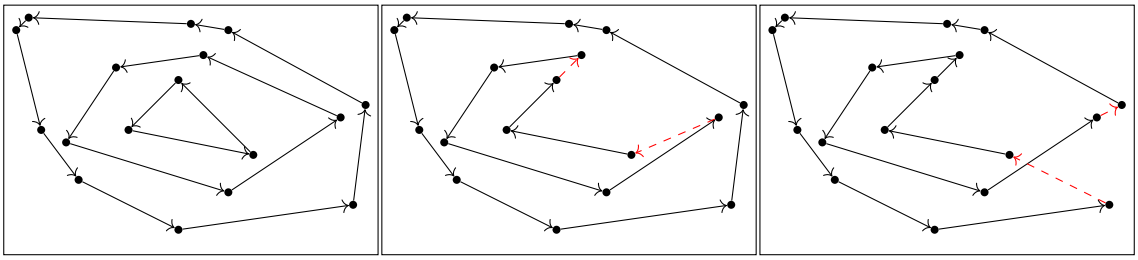


Figure 2: Example of constructing an initial solution for the AngleTSP.

3.1.2 Initial solution for the AngleDistanceTSP

By plotting several optimal solutions of the AngleDistanceTSP in which the turning angle and distance component are quite balanced, we observed that optimal tours often contain no intersections. To obtain such a closed path with no intersections, we designed a construction heuristic that has similarities to the sweep algorithm of Gillett and Miller (1974). We first select a starting vertex v_0. Then, the remaining vertices in $V \setminus \{v_0\}$ are sorted in increasing order by comparing the polar angles via cross products (see above). This results in a list of vertices arranged in counterclockwise order. Finally, we obtain a tour by connecting v_0 with the first vertex of the list, all the vertices in the list according to the order in which they appear, and the last vertex of the list to v_0. Because the resulting tour depends on the choice of the starting vertex, we repeat the procedure for all possible starting vertices, and we choose the tour with the lowest cost. Figure 3 shows an example of four tours obtained by selecting different starting vertices highlighted in red.

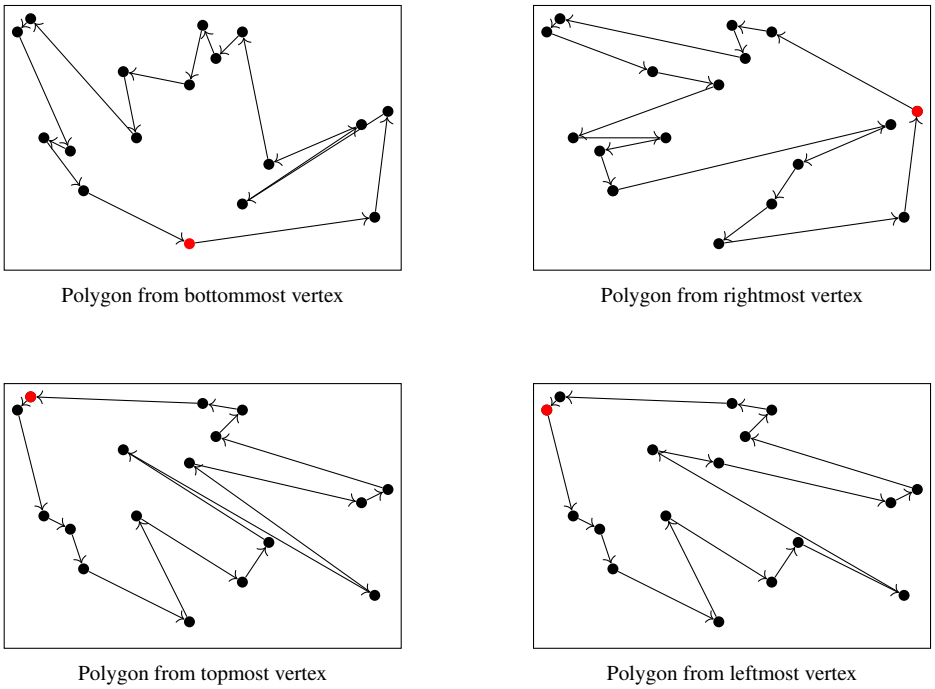


Figure 3: Example of four tours obtained from different starting vertices for the AngleDistanceTSP.

3.2 Neighborhoods

In each iteration, GTS-angular applies the best non-tabu move in a composite neighborhood defined by the set $\mathscr{O}$ containing three operators: relocate, exchange, and 2-opt (Toth and Vigo 2003). Differently from the standard TSP, to compute the impact that each move has on the cost of a solution, we have to take into account that the cost is defined for triplets of vertices. This implies that each modification of a vertex position causes a change not only of the cost corresponding to that vertex but also on the cost associated with its predecessor and successor.

All operators are defined using the generator arc principle introduced in Section 3. Figure 4 presents the operators:

- The relocate operator moves one vertex from its current position to another one in the tour.
- The exchange operator swaps the positions of two vertices in the tour.
- The 2-opt operator removes two non-consecutive arcs from the tour, inverts the vertex segment between the two removed arcs, and reconnects the inverted segment using two new arcs.

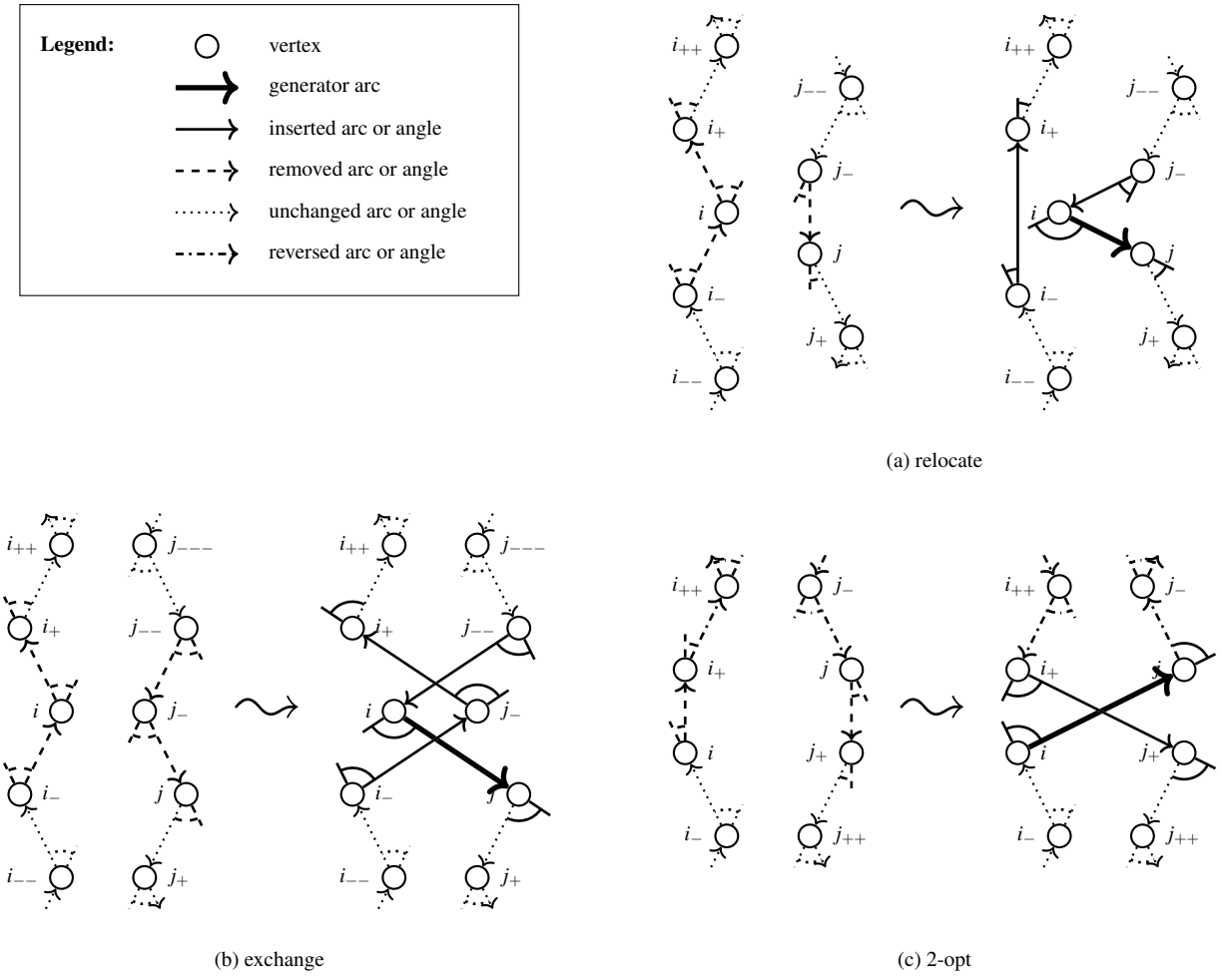


Figure 4: Neighborhood operators of GTS-angular. The generator arc is denoted by (i, j) and highlighted in bold. The predecessor and successor of i are denoted by i_- and i_+, respectively.

3.3 Construction of the generator arc set

To reduce the neighborhood size, we recall that the generator arc set A_g contains a subset of the complete arc set A and that this subset is determined by applying sparsification methods. Sparsification methods were introduced by Toth and Vigo (2003) and have later been used in multiple works (see, e.g., Prins et al. 2007, Escobar et al. 2014, Goeke 2019).

In our case, A_g is composed of the arcs contained in the arc sets A_a and A_s. A_a includes the arcs accepted by the most recent moves, and we explain how it is obtained in Section 3.3.1. A_s contains the arcs selected by one of our two sparsification strategies, and, in Section 3.3.2, we describe how it is derived. Finally, in Section 3.3.3, we provide the details on how the arcs in A_a and A_s are merged into set A_g.

3.3.1 Composition of A_a

Different strategies have been suggested for guaranteeing that the generator arc set contains the arcs belonging to high-quality solutions. Toth and Vigo (2003), Schneider et al. (2017), and Becker et al. (2021) include the arcs that were inserted in the best move selected in each iteration. In the first two papers, the authors remove these arcs repetitively after a given number of iterations, while, in the third paper, these arcs are always kept. Neverthess, all these papers agree that the inclusion of the arcs inserted by the best move in the last iteration is fundamental for achieving good solution quality.

In GTS-angular, we add the arcs inserted by the most recently accepted moves to the generator arc set. Initially, A_a is an empty set, and, during the search, its cardinality cannot exceed the limit $\lceil \kappa_a |A| \rceil$, with $0 < \kappa_a < 1$. At the end of each iteration, after the move is carried out, the set of arcs inserted by the applied move is represented by A_{ins}, and each of its elements is added to A_a with a time stamp representing the moment in which they have been inserted. If, in an iteration, an arc in A_{ins} has already been added by previous accepted moves, the time stamp of that arc is updated. When the cardinality limit of A_a is reached, the arcs with the oldest time stamp are removed from the set and those added by the last move are inserted.

3.3.2 Composition of A_s

Even if most of the papers using sparsification strategies rely on the arc cost-based sparsification method originally proposed by Toth and Vigo (2003), other authors have presented alternative methodologies (Labadie et al. 2012, Schneider et al. 2017). Due to the special features of the AngleTSP and AngleDistanceTSP, we propose two sparsification strategies.

Both approaches rely on a modification of the general lens procedure (LENS) introduced by Staněk et al. (2019). In LENS, a lens with thickness γ is positioned between two consecutive vertices i and i_+ in a tour such that the lens curvature intersects at these vertices. Only the vertices inside this lens are considered for insertion in the tour between i and i_+. This procedure guarantees that any inserted vertex generates: (1) an additional angle that is smaller than the angle of the lens curvature at the tangent point with the vertices, and (2) a limited increase in the traveled distance. The left part of Figure 5 shows an example of LENS applied to the arc (i, i_+). In this example, only vertex k is within the lens and hence considered for insertion between i and i_+. The right part of Figure 5 shows the resulting insertion and updated tour section.

In contrast to Staněk et al. (2019), who apply LENS to every arc of a tour and include all the resulting arcs in A_s, we further intensify the sparsification by limiting the cardinality of A_s to $\lceil \kappa_s |A| \rceil$, with $0 < \kappa_s < 1$.

Figure 5: Example of LENS applied to the arc (i, i_+).

The parameter κ_s represents the sparsification intensity: the lower κ_s, the more intense the sparsification, i.e., only a few arcs are included in A_s. This implies that a criterion to select the arcs on which the lens should be positioned has to be defined. To this end, we propose the following two strengthened LENS-based sparsification strategies:

- **Random-based lens procedure (r-LENS)**: In each iteration, the lens is initially positioned on the arc originating from a vertex i, and then applied to the subsequent arcs according to their order in the tour. However, across iterations, if the lens is always initially positioned on the arc originating from the same vertex i, there may be little variety in A_s due to its restricted cardinality. To prevent this, r-LENS starts positioning the lens on the arc of a random vertex in each iteration. This ensures that different parts of the tour are targeted across iterations.
- **Cost-based lens procedure (c-LENS)**. According to this approach, the turning angles of the tour are first sorted in non-increasing order. Then, the lens is applied sequentially, from the pair of arcs defining the largest angle to the smallest one, until the cardinality limit of A_s is reached. This sparsification procedure is greedier than r-LENS, and it aims to improve the most expensive parts of the tour.

Contrary to the sparsification methods proposed in the literature, which are applied only once before the search starts, our sparsification methods depend on the current solution and are therefore called at the end of each iteration of GTS-angular, and applied to the tour resulting after carrying out the best non-tabu move. Arcs which have already been included in A_a are not added to A_s.

To introduce more diversification, both sparsification strategies are dynamic. In contrast to common practice in the literature, in which the dynamic component is based on variations of the sparsification intensity κ_s (e.g., Toth and Vigo 2003, Schneider and Löffler 2019), in r-LENS and c-LENS, the sparsification intensity remains unchanged. Instead, we modify the criterion according to which the arcs are selected, i.e., we vary the lens thickness in the interval $[\gamma_{min}, 2\gamma_{min}]$. More precisely, the lens thickness is initially set equal to γ_{min}. Every η^κ iterations without improvement, the lens thickness is increased by γ_{min}/μ, where $\mu = \eta/\eta^\kappa$-1. In the last η^κ iterations of GTS-angular, the lens thickness is doubled with respect to its initial value γ_{min}. Enlarging the lens thickness γ when improvements are hard to find allows inserting more diverse arcs in the generator arc set while keeping its size and the resulting neighborhood restricted. Whenever an improvement with respect to the overall best solution is found, the lens thickness is reset to γ_{min}.

Based on the used sparsification method, we derive two algorithmic variants, namely GTS-angular r-LENS and GTS-angular c-LENS.

3.3.3 Composition of A_g

Figure 6 provides an overview of how the generator arc set A_g is updated during the search depending on the sets A_a and A_s. At the beginning of GTS-angular, A_s contains the arcs added by the sparsification strategy

applied to the tour of the current solution S^{curr} resulting from the construction heuristic, and A_a and A_g are initialized to empty sets (line 1). All arcs contained in A_s are added to A_g (line 2). Then, lines 4–19 of the pseudocode in Figure 1 are executed. At the end of every iteration of GTS-angular (see Figure 1), after the best move is carried out, the tour of the new current solution S^{curr} and the set A_{ins} of the arcs inserted by the move are obtained (line 4). Next, A_g is cleaned from all arcs (line 5). The set A_a is updated by adding the arcs contained in A_{ins} (line 6). Subsequently, A_s is obtained by applying the sparsification strategy to the tour of the current solution S^{curr} and by discarding those arcs which are already in A_a (line 7). If these arcs would be included in A_s, they would increase the cardinality of A_s unnecessarily (and consequently prevent other arcs to be added to A_s) because they would be part of A_g in any case. Finally, the arcs contained in A_a and A_s are merged into A_g (line 8). The set A_g is then used for the next iteration of GTS-angular.

Pseudocode for the composition of the generator arc set A_g.

```
1 initialize A_s ← sparsificationStrategy(S^curr), A_a = ∅, A_g = ∅
2 Add arcs in A_s to A_g
3 while termination criterion not satisfied do
4 |   S^curr, A_ins ← apply best move {lines 4–19 of pseudocode in Figure 1}
5 |   Delete all arcs from A_g
6 |   Add arcs in A_ins to A_a
7 |   A_s ← sparsificationStrategy(S^curr, A_a)
8 |   Add arcs in A_a and A_s to A_g
9 end
```

Figure 6: Pseudocode for the composition of the generator arc set A_g.

3.4 Continuous diversification

We incorporate the continuous diversification strategy originally proposed by Cordeau et al. (1997) into GTS-angular. This diversification technique has been successfully applied in a number of TS metaheuristics developed for different problem classes (e.g., the CLRP and the multi-depot VRP). The goal of continuous diversification is to guide the search towards unexplored regions of the solution space.

Whenever a move does not improve the cost value, i.e., a move is changing S into $S^{'}$ with $C(S^{'}) \geq C(S)$, we penalize the cost of that solution based on its similarity with respect to the previously visited solutions. To measure this similarity, we keep a counter $\Lambda(a)$ for every arc $a \in A$. Every time GTS-angular carries out a move that inserts the set of arcs A_{ins}, we increase $\Lambda(a)$ by one for every arc $a \in A_{ins}$. The resulting formula to evaluate non-improving solutions is the following:

$$C_{div}(S^{'}) = C(S^{'}) \left(1 + \sigma \cdot \sum_{a \in A_{ins}} \Lambda(a) \right), \tag{1}$$

where σ is a parameter that controls the intensity of the diversification. Whenever a new best solution is found, $\Lambda(a)$ is reset to zero for every arc $a \in A$; thus, $C_{div}(S^{'}) = C(S^{'})$.

4 Computational experiments

The goal of the computational experiments is to compare our two algorithmic variants (GTS-angular r-LENS and GTS-angular c-LENS) to the state-of-the-art algorithms from the literature.

Section 4.1 presents the instances on which the two algorithmic variants are tested and the computational environment. In Section 4.2, we describe the parameter setting for GTS-angular. Finally, Section 4.3 presents the comparison of the results to the state-of-the-art heuristics.

4.1 Benchmark instances and computational environment

To test the performance of GTS-angular, we run computational experiments on the benchmark set proposed by Staněk et al. (2019) and available at https://arxiv.org/abs/1803.03681. This set is composed of ten symmetric instances for each of the 40 different sizes $n = 5, 10, \ldots, 200$. In total, 400 instances for the AngleTSP and 400 instances for the AngleDistanceTSP are considered. Each AngleDistanceTSP instance has been obtained taking the same vertex coordinates as the corresponding AngleTSP instance. The difference between the two instance sets lies in how the cost of the AngleTSP and of the AngleDistanceTSP instance is computed. In the AngleTSP, the cost for each triplet of vertices $i, j, k \in V$ corresponds to the turning angle α_{ijk} multiplied by 1000. In the AngleDistanceTSP, the cost for each triplet of vertices $i, j, k \in V$ is obtained by combining the turning angle and the Euclidean distance as follows:

$$c_{ijk} = 100\left(40 \cdot \alpha_{ijk} + \frac{d_{ij} + d_{jk}}{2}\right). \tag{2}$$

In both cases, the resulting costs are rounded to 12 decimal places.

GTS-angular was implemented in C++ and compiled using Clang version 12.0.0. The experiments were performed on an Intel(R) Xeon(R) computer with a CPU E5-2430 v2 processor, at 2.50GHz with 64 GB RAM under CentOS GNU/Linux 7. Because our algorithm contains randomized elements, we performed ten runs for each instance.

4.2 Parameter setting

During the development of the algorithm, parameter tuning experiments have been performed to adequately set the parameters for GTS-angular. To avoid overfitting the parameters to the instance type, we set all GTS-angular parameters to the same values for solving both the AngleTSP and the AngleDistanceTSP instances, except for the minimum lens thickness parameter γ_{min}. Because of the different computation of the costs for each triplet of vertices performed in the AngleTSP and AngleDistanceTSP instances (see Equation 2), a different value for the minimum lens thickness for each problem type, namely γ^{A}_{min} for the AngleTSP and γ^{AD}_{min} for the AngleDistanceTSP, should be properly tuned. Consequently, we considered the same lens thickness value of 0.6981 suggested by Staněk et al. (2019) for solving the AngleTSP instances and, consistently with the computation of the turning angle (see Section 4.1), we multiply that by 1000, that is, γ^{A}_{min} =698.1. For the AngleDistanceTSP instances, we computed γ^{AD}_{min} consistently to Equation (2) as γ^{AD}_{min} =$100(40 \cdot 0.6981 + \overline{d})$, where $\overline{d}$ is the average distance over all arcs in A. Table 1 summarizes all GTS-angular parameters.

4.3 Comparison to the literature

In this section, we compare the performance of GTS-angular r-LENS and GTS-angular c-LENS to the state-of-the-art heuristics from the literature. For this comparison, we consider the results of Staněk et al.

Component	Notation	Parameter values
tabu tenure	$\tau = [\tau_{min}, \tau_{max}]$	$\tau = [10, 30]$
termination criterion	η	$\eta = 30\,000$
continuous diversification	σ	$\sigma = 2e^{-6}$
dynamic sparsification	$\gamma = [\gamma^{A}_{min}, 2\gamma^{A}_{min}]$	$\gamma = [698.1, 2 \cdot 698.1] = [698.1, 1\,396.2]$
	$\gamma = [\gamma^{AD}_{min}, 2\gamma^{AD}_{min}]$	$\gamma = [100(40 \cdot 0.6981 + \overline{d}), 200(40 \cdot 0.6981 + \overline{d})]$
		$= [100(2\,792.4 + \overline{d}), 200(2\,792.4 + \overline{d})]$
accepted arc list length limit	κ_a	$\kappa_a = 0.05$
sparsification intensity	κ_s	$\kappa_s = 0.05$
sparsification iterations	η^{κ}	$\eta^{\kappa} = 1\,000$

Table 1: GTS-angular parameter values.

(2019), that include:

- The solution (objective function value and runtime) obtained by each of their algorithmic variants for each instance.
- The optimal objective function value for each of the AngleTSP instances with size up to 75 vertices, and for each of the AngleDistanceTSP instances with size up to 100 vertices. The number of AngleTSP and AngleDistanceTSP instances for which an optimal solution is available is reported in Table 2. These optimal solutions are obtained by solving the integer linear models for the AngleTSP and AngleDistanceTSP via an off-the-shelf solver (for more details, see Staněk et al. 2019).

	Number of instances	
	Optimal solution available	Optimal solution not available
AngleTSP	150	250
AngleDistanceTSP	200	200
Total	350	450

Table 2: Number of AngleTSP and AngleDistanceTSP instances for which an optimal solution is available from Staněk et al. (2019).

The best-known solution (BKS) corresponds to the optimal objective function value for the instances for which an optimal solution is available, and to the lowest objective function value across all the algorithmic variants of Staněk et al. (2019) for those instances for which optimality is not proven.

For each algorithmic variant of Staněk et al. (2019), we first determine whether it is dominated or not (see Tables 5 and 6 in Appendix A). Specifically, we compute, for each algorithmic variant the average gap (over all instances) between the obtained objective function value and the BKS and the average runtimes. The status of an algorithm is "not dominated" if there is no other algorithm that provides a better solution quality and faster runtimes simultaneously. Otherwise, it is "dominated". For the comparison with GTS-angular, we consider only the non-dominated algorithmic variants that show an average gap to the BKS below 2%. This results in the selection of four algorithmic variants (namely, $LPP^{R} + M^{15}$, $LPP^{R} + M^{20}$, $LPC^{R}_{1} + M^{15}$, $LPC^{R}_{1} + M^{20}$) for the AngleTSP, and six algorithmic variants (namely, $CIF + M^{15}$, $CIF + M^{20}$, $NN^{2}_{S} + M^{15}$, $NN^{2}_{S} + M^{20}$, $LPC^{R}_{1} + M^{20}$, $LPC^{R}_{2} + M^{20}$) for the AngleDistanceTSP. The algorithmic variants with names starting with "*LP*" are methods based on the solution of the linear programming relaxation of the AngleTSP

and AngleDistanceTSP models. They differ with respect to the rounding procedure applied to the variables representing the selection of the arcs in a tour. The algorithmic variants with names starting with "*CIF*" are methods based on the cheapest insertion heuristic proposed by Fischer et al. (2014). The algorithmic variants with names starting by "NN_S^2" are methods based on the nearest neighbor heuristic run from all possible starting vertices and coupled with the 2-opt improvement heuristic. The suffixes "M^{15}" and "M^{20}" refer to an improvement heuristic similar to a large neighborhood search, in which part of the tour is destroyed by removing all arcs adjacent to a set of vertices in a geographical neighborhood. The numbers 15 and 20 represent the expected number of vertices in such a set. Then, the tour is repaired by optimally reconnecting the paths and isolated points by solving a quadratic programming model.

To allow for a fair runtime comparison to the results from the literature, which are obtained using a computer with a different CPU, we used the CPU mark defined by PassMark Software (2021). This website assigns a score of 6864 to our CPU, and a score of 7207 to the Intel(R) Core(TM) i7-4790 CPU @ 3.60 GHz used by Staněk et al. (2019). Thus, our CPU is 0.95 times faster (i.e., because the conversion factor is below one, it is slower), and we convert all our runtimes by multiplying them by 0.95 for fair comparison. We are aware that a completely precise comparison of runtimes is not possible due to the different programming languages. However, we follow the procedure adopted by the literature, and we compare with respect to the CPU.

Tables 3 and 4 summarize the comparison of GTS-angular r-LENS and GTS-angular c-LENS for the AngleTSP and AngleDistanceTSP, respectively, to the non-dominated algorithms of Staněk et al. (2019). For each solution method, we report the gap of the best run to the BKS averaged over all instances ($\overline{\Delta}_{BKS}^{best}(\%)$), the gap to the BKS averaged over all runs and instances ($\overline{\Delta}_{BKS}^{avg}(\%)$), and the runtimes $\bar{t}(s)$ averaged over all runs and instances. In the tables, we highlight in bold the best result for each of these three measures. Tables 7–10 in Appendix B contain more detailed results grouped according to the instance size. In Figure 7, we represent the Pareto frontiers of the algorithms with respect to the average runtime $\bar{t}(s)$ and the average gap to the BKS $\overline{\Delta}_{BKS}^{avg}(\%)$ for the AngleTSP and AngleDistanceTSP, respectively. Every time the performance of the algorithms is discussed in terms of solution quality and runtimes, we use the average solution quality and average runtimes of GTS-angular for fair comparison. The solution of the best run is only used to carry out comparisons based exclusively on solution quality.

On AngleTSP instances, GTS-angular r-LENS, $LPP^R + M^{20}$, and $LPC_1^R + M^{20}$ dominate GTS-angular c-LENS. The position of GTS-angular r-LENS on the Pareto frontier (Figure 7a) shows that GTS-angular r-LENS achieves a good compromise between solution quality and runtime.

For the AngleDistanceTSP, both GTS-angular variants achieve a better solution quality than the algorithms of Staněk et al. (2019) in competitive runtimes. GTS-angular r-LENS returns a better solution quality than GTS-angular c-LENS, while the latter is faster. The position of GTS-angular c-LENS in Figure 7b shows that it lies not only on the Pareto frontier, but it also dominates two state-of-the-art algorithms (namely, $LPC_1^R + M^{20}$ and $LPC_2^R + M^{20}$).

Analyzing the results with regard to the average gap to the BKS obtained in the best run, i.e., $\overline{\Delta}_{BKS}^{best}(\%)$, and reported in Tables 3 and 4, our algorithmic variants achieve a better solution quality than the algorithms of Staněk et al. (2019) both for the AngleTSP and AngleDistanceTSP instances. Specifically, GTS-angular r-LENS finds new BKSs for 177 AngleTSP instances, i.e., 70.8% of the instances for which an optimal solution is not available (see Table 2), and for 183 AngleDistanceTSP instances, i.e., 91.5% of the instances for which an optimal solution is not available. Moreover, the heuristics of Staněk et al. (2019) returning the best solution quality (i.e., $LPC_1^R + M^{20}$ for AngleTSP instances, and $LPC_2^R + M^{20}$ for AngleDistanceTSP in-

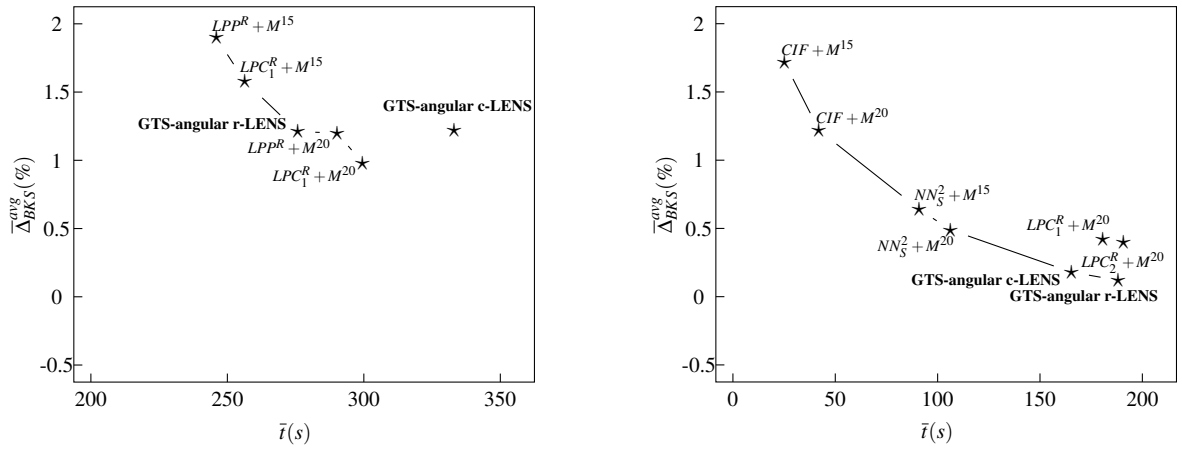
stances) reported, for instances with cardinality between 105 and 200, average gaps to the best lower bounds of 15.75% and 8.26% for the AngleTSP and AngleDistanceTSP, respectively. For the same instances, our average gaps to the best lower bounds are 13.89% for the AngleTSP instances and 7.30% for the AngleDistanceTSP instances.

	$LPP^R + M^{15}$	$LPP^R + M^{20}$	$LPC_1^R + M^{15}$	$LPC_1^R + M^{20}$	GTS-angular r-LENS	GTS-angular c-LENS
$\overline{\Delta}_{BKS}^{best}(\%)$	1.90	1.20	1.58	0.98	**-0.36**	-0.13
$\overline{\Delta}_{BKS}^{avg}(\%)$	1.90	1.20	1.58	**0.98**	1.28	1.29
$\bar{t}(s)$	**245.95**	290.26	256.32	299.53	275.77	333.12

Table 3: AngleTSP results: comparison of the average results of the non-dominated algorithms of Staněk et al. (2019) with GTS-angular r-LENS and GTS-angular c-LENS.

	$CIF + M^{15}$	$CIF + M^{20}$	$NN_S^2 + M^{15}$	$NN_S^2 + M^{20}$	$LPC_1^R + M^{20}$	$LPC_2^R + M^{20}$	GTS-angular r-LENS	GTS-angular c-LENS
$\overline{\Delta}_{BKS}^{best}(\%)$	1.72	1.22	0.64	0.49	0.42	0.40	**-0.20**	-0.11
$\overline{\Delta}_{BKS}^{avg}(\%)$	1.72	1.22	0.64	0.49	0.42	0.40	**0.12**	0.18
$\bar{t}(s)$	**25.13**	42.00	90.97	106.27	180.77	190.72	188.18	165.33

Table 4: AngleDistanceTSP results: comparison of the average results of the non-dominated algorithms of Staněk et al. (2019) with GTS-angular r-LENS and GTS-angular c-LENS.



(a) AngleTSP. (b) AngleDistanceTSP.

Figure 7: Pareto frontiers of the compared algorithmic variants.

To provide more details on our comparison, we analyze the following indicators computed by grouping instances by size: (1) the percentage of instances for which each of the algorithms is able to find the optimal solution (Section 4.3.1), and (2) the percentage of instances on which our GTS-angular variants provide better or equal solution quality than the one provided by the best-performing heuristics of Staněk et al. (2019) (Section 4.3.2). For the sake of representation, the figures in the remainder of this section show aggregated data. Specifically, we first obtain groups of instances by dividing them according to their size aggregated in increasing size intervals of 20. For example, the instance group 5–25 contains all instances with size between 5 and 25 vertices, the instance group 30–50 contains all instances with size between 30 and 50 vertices, and so on. Then, we compute the percentage of instances within each group satisfying the evaluated indicator, and we report that value.

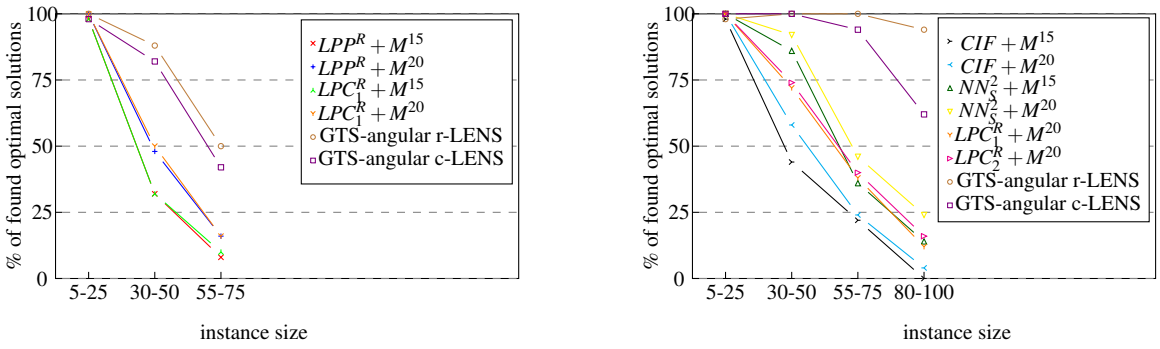
4.3.1 Percentage of optimal solutions found

According to this indicator, we compute the percentage of instances for which our GTS-angular variants and the heuristics proposed by Staněk et al. (2019) are able to find the optimal solution. To derive this indicator, we refer to the optimal objective function values computed by Staněk et al. (2019). We recall that these optimal values were provided only for the instances with sizes 5–75 for the AngleTSP, and instances with sizes 5–100 for the AngleDistanceTSP. Because their heuristics do not contain random components, the authors executed only one run for each heuristic. Hence, we compute this indicator for their algorithms by using the objective function value of that run. For our GTS-angular variants, we consider the objective function value of the best run. Figure 8 shows, for each heuristic, the percentage of AngleTSP and AngleDistanceTSP instances for which the optimal solution is found.

For the AngleTSP instances (Figure 8a), GTS-angular r-LENS and GTS-angular c-LENS are the heuristics returning the highest percentage of optimal solutions. For instances with size ranging between 30 and 75 vertices, our algorithmic variants always find at least 25% more optimal solutions compared with the variants of Staněk et al. (2019).

For the AngleDistanceTSP instances (Figure 8b), our GTS-angular variants find optimal solutions for most of the instances with up to 100 vertices. Conversely, the heuristics of Staněk et al. (2019) are not able to find optimal solutions for most of the instances with sizes starting from 55 vertices.

Comparing our two algorithmic variants, GTS-angular r-LENS is in general superior to GTS-angular c-LENS. An exception is represented by the case of AngleDistanceTSP instances with sizes ranging from 5 to 25. In this instance group, GTS-angular c-LENS performs slightly better.



(a) AngleTSP. (b) AngleDistanceTSP.

Figure 8: Percentage of optimal solutions found by all compared heuristics.

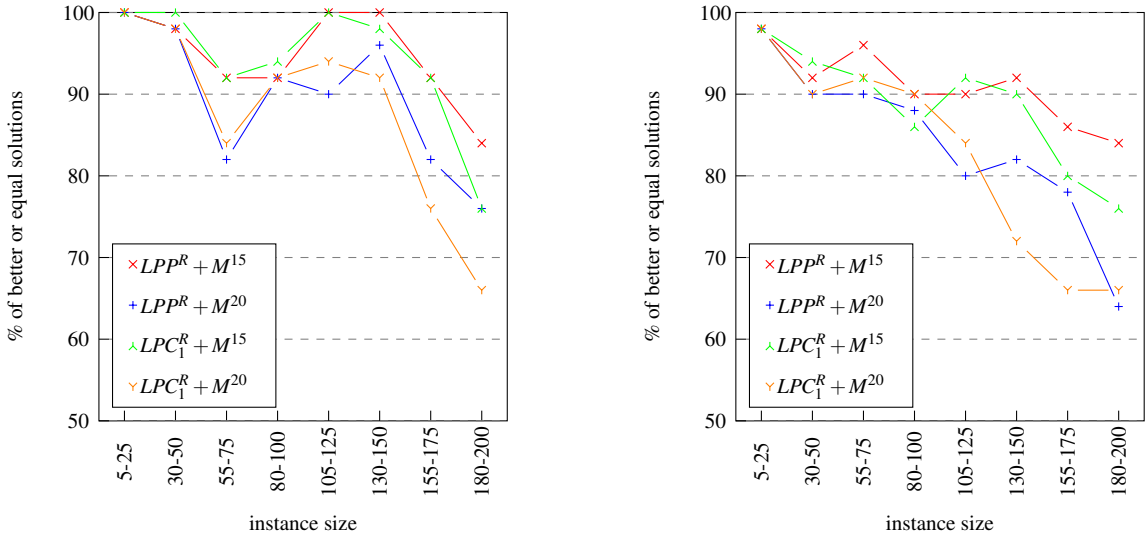
4.3.2 Percentage of instances with better or equal solution quality

According to this indicator, we compute the percentage of instances for which our GTS-angular variants find a better or equal solution than the one provided by each of the non-dominated heuristics of Staněk et al. (2019). For each of our algorithmic variants, we count the number of instances for which the returned objective function value of the best run is lower or equal to the one provided by each of the non-dominated heuristics of Staněk et al. (2019). Figures 9 and 10 show the percentage of these instances for GTS-angular r-LENS and GTS-angular c-LENS for AngleTSP and AngleDistanceTSP instances, respectively.

For the AngleTSP, both GTS-angular variants (Figures 9a and 9b) return better or equal solutions than the non-dominated variants of Staněk et al. (2019) for more than 60% of the instances, independent of the

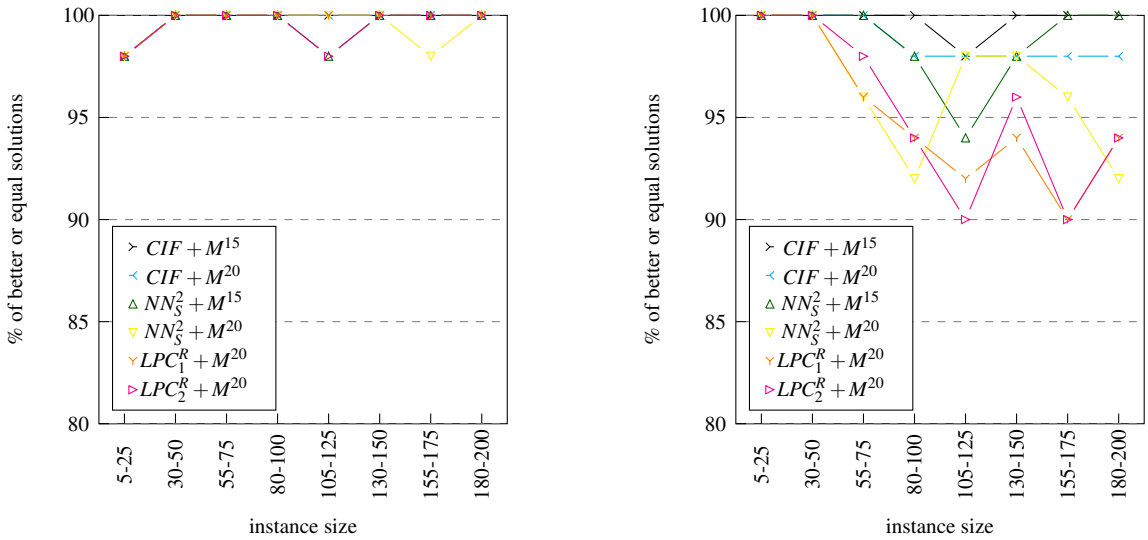
instance size. Compared to $LPC_1^R + M^{20}$ (the best-performing heuristic with respect to solution quality by Staněk et al. 2019), GTS-angular r-LENS finds better or equal solutions for more than 75% of the instances with up to 175 vertices and for more than 65% of the instances with 180 to 200 vertices.

For the AngleDistanceTSP, the performance of our GTS-angular variants is convincing: for all but three instance size groups, GTS-angular r-LENS finds better or equal solutions for all instances. In those three instance size groups, GTS-angular r-LENS returns better or equal solutions for more than 97% of the instances (Figure 10a). Across all instance size groups, GTS-angular c-LENS finds better or equal solutions for at least 90% of the instances (Figure 10b). In addition, GTS-angular r-LENS always returns a better or equal solution than the one found by the algorithms of Staněk et al. (2019) for the biggest instances with sizes 180-200.



(a) GTS-angular with r-LENS. (b) GTS-angular with c-LENS.

Figure 9: Percentage of AngleTSP instances for which GTS-angular finds a better or equal solution than the ones of the heuristics of Staněk et al. (2019).



(a) GTS-angular with r-LENS. (b) GTS-angular with c-LENS.

Figure 10: Percentage of AngleDistanceTSP instances for which GTS-angular finds a better or equal solution than the ones of the heuristics of Staněk et al. (2019).

5 Conclusion

We propose a GTS framework that considers the geometric features of the AngleTSP and AngleDistanceTSP in the design of the starting solutions and the sparsification methods. The comparison of two sparsification methods proves that r-LENS returns solutions of better quality than c-LENS on AngleTSP and AngleDistanceTSP instances, but c-LENS is faster on AngleDistanceTSP instances. The computational results on AngleTSP instances show that the performance of GTS-angular lies on the Pareto frontier of heuristic AngleTSP and AngleDistanceTSP methods and it qualifies as a valuable alternative to the heuristics proposed by Staněk et al. (2019). For the AngleDistanceTSP instances, GTS-angular provides the best solution quality across the best-performing heuristics of Staněk et al. (2019) in competitive runtimes. Moreover, it dominates two of the heuristics proposed by Staněk et al. (2019). Finally, it finds new best-known solutions on around 81% of AngleTSP and AngleDistanceTSP instances for which an optimal solution is not available.

References

A. Aggarwal, D. Coppersmith, S. Khanna, R. Motwani, and B. Schieber. The angular-metric traveling salesman problem. *SIAM Journal on Computing*, 29(3):697–711, 2000.

O. Aichholzer, A. Fischer, F. Fischer, J. F. Meier, U. Pferschy, A. Pilz, and R. Staněk. Minimization and maximization versions of the quadratic travelling salesman problem. *Optimization*, 66(4):521–546, 2017.

C. Becker, R. Cavagnini, S. Irnich, and M. Schneider. Rule-based design of a granular tabu search for the multi-depot vehicle routing problem. Working Paper DPO-2021-01, Deutsche Post Chair – Optimization of Distribution Networks, RWTH Aachen University, Aachen, Germany, 2021.

J.-F. Cordeau, M. Gendreau, and G. Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2):105–119, 1997.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, 2009.

J. W. Escobar, R. Linfati, P. Toth, and M. G. Baldoquin. A hybrid granular tabu search algorithm for the multi-depot vehicle routing problem. *Journal of Heuristics*, 20(5):483–509, 2014.

A. Fischer and C. Helmberg. The symmetric quadratic traveling salesman problem. *Mathematical Programming*, 142 (1):205–254, 2013.

A. Fischer, F. Fischer, G. Jäger, J. Keilwagen, P. Molitor, and I. Grosse. Exact algorithms and heuristics for the quadratic traveling salesman problem with an application in bioinformatics. *Discrete Applied Mathematics*, 166: 97–114, 2014.

A. Fischer, F. Fischer, G. Jäger, J. Keilwagen, P. Molitor, and I. Grosse. Computational recognition of RNA splice sites by exact algorithms for the quadratic traveling salesman problem. *Computation*, 3(2):285–298, 2015.

B. E. Gillett and L. R. Miller. A heuristic algorithm for the vehicle-dispatch problem. *Operations research*, 22(2): 340–349, 1974.

F. Glover. Tabu search – part 1. *ORSA Journal on Computing*, 1(3):190–206, 1989.

D. Goeke. Granular tabu search for the pickup and delivery problem with time windows and electric vehicles. *European Journal of Operational Research*, 278(3):821–836, 2019.

R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1:132–133, 1972.

D. Kirchler and R. Wolfler Calvo. A granular tabu search algorithm for the dial-a-ride problem. *Transportation Research Part B*, 56:120–135, 2013.

N. Labadie, R. Mansini, J. Melechovskỳ, and R. W. Calvo. The team orienteering problem with time windows: An LP-based granular variable neighborhood search. *European Journal of Operational Research*, 220(1):15–27, 2012.

A. C. Medeiros and S. Urrutia. Discrete optimization methods to determine trajectories for Dubins' vehicles. *Electronic Notes in Discrete Mathematics*, 36:17–24, 2010.

PassMark Software. Professional CPU benchmarks, 2021. URL https://www.cpubenchmark.net/singleThread.html.

C. Prins, C. Prodhon, A. Ruiz, P. Soriano, and R. Wolfler Calvo. Solving the capacitated location-routing problem by a cooperative Lagrangean relaxation-granular tabu search heuristic. *Transportation Science*, 41(4):470–483, 2007.

K. Savla, E. Frazzoli, and F. Bullo. Traveling salesperson problems for the dubins vehicle. *IEEE Transactions on Automatic Control*, 53(6):1378–1391, 2008.

M. Schneider and M. Löffler. Large composite neighborhoods for the capacitated location-routing problem. *Transportation Science*, 53(1):301–318, 2019.

M. Schneider, F. Schwahn, and D. Vigo. Designing granular solution methods for routing problems with time windows. *European Journal of Operational Research*, 263(2):493–509, 2017.

R. Staněk, P. Greistorfer, K. Ladner, and U. Pferschy. Geometric and LP-based heuristics for angular travelling salesman problems in the plane. *Computers & Operations Research*, 108:97–111, 2019.

P. Toth and D. Vigo. The granular tabu search and its application to the vehicle-routing problem. *INFORMS Journal on computing*, 15(4):333–346, 2003.

Appendix

A Dominated and non-dominated algorithmic variants by Staněk et al. (2019)

Tables 5 and 6 report the dominated and non-dominated algorithmic variants proposed by Staněk et al. (2019) for the AngleTSP and AngleDistanceTSP instances, respectively.

Algorithmic variant	$\overline{\Delta}_{BKS}(\%)$	$\bar{t}(s)$	dominated/not dominated
CIF	56.5	3.49	dominated
$CIF+2O$	22.75	3.85	dominated
$CIF+L$	48.89	43.7	dominated
$CIF+M^{15}$	7.67	66.19	dominated
$CIF+M^{20}$	5.33	104.65	dominated
$CIF+3O$	11.74	88.61	dominated
NN_S	39.57	146.51	dominated
NN_S+2O	35.92	146.64	dominated
NN_S+L	38.6	187.77	dominated
NN_S+M^{15}	12.3	253.81	dominated
NN_S+M^{20}	8.64	317.54	dominated
NN_S+3O	26.93	215.02	dominated
NN_S^L	36.03	215.33	dominated
NN_S^L+2O	19.49	215.62	dominated
NN_S^L+L	32.06	254.86	dominated
$NN_S^L+M^{15}$	3.97	278.38	dominated
$NN_S^L+M^{20}$	2.59	329.23	dominated
NN_S^L+3O	9.0	296.41	dominated
NN_S^2	28.93	35.61	dominated
NN_S^2+2O	28.93	35.67	dominated
NN_S^2+L	28.02	76.84	dominated
$NN_S^2+M^{15}$	11.25	138.75	dominated
$NN_S^2+M^{20}$	7.91	198.77	dominated
NN_S^2+3O	23.89	102.17	dominated
CH	41.07	0.81	dominated
$CH+2O$	39.35	0.92	dominated
$CH+L$	40.28	42.2	dominated
$CH+M^{15}$	12.22	106.32	dominated
$CH+M^{20}$	8.98	175.15	dominated
$CH+3O$	29.13	72.91	dominated
CH_C	35.87	1.22	dominated
CH_C+2O	35.45	1.3	dominated
CH_C+L	35.48	42.63	dominated
CH_C+M^{15}	12.72	109.7	dominated
CH_C+M^{20}	8.95	176.71	dominated
CH_C+3O	29.97	59.39	dominated
CH_C^L	32.9	0.73	not dominated
CH_C^L+2O	21.76	0.98	not dominated
CH_C^L+L	29.64	40.34	dominated
$CH_C^L+M^{15}$	3.92	66.17	not dominated
$CH_C^L+M^{20}$	2.68	120.13	not dominated
CH_C^L+3O	9.01	86.96	dominated
LPP	25.69	107.78	dominated
$LPP+2O$	13.25	108.05	dominated
$LPP+L$	22.15	148.14	dominated
$LPP+M^{15}$	4.18	162.68	dominated
$LPP+M^{20}$	2.65	213.59	not dominated
$LPP+3O$	6.43	178.42	dominated
LPP^R	10.39	192.41	dominated
LPP^R+2O	7.2	192.57	dominated
LPP^R+L	8.66	233.06	dominated
LPP^R+M^{15}	1.9	245.95	not dominated
LPP^R+M^{20}	1.2	290.26	not dominated
LPP^R+3O	4.29	252.79	dominated
LPC_1^R	8.56	204.32	dominated
LPC_1^R+2O	6.44	204.46	dominated
LPC_1^R+L	7.17	244.93	dominated
$LPC_1^R+M^{15}$	1.58	256.32	not dominated
$LPC_1^R+M^{20}$	0.98	299.53	not dominated
LPC_1^R+3O	3.71	262.42	dominated
LPC_2^R	8.38	212.48	dominated
LPC_2^R+2O	6.4	212.62	dominated
LPC_2^R+L	6.99	253.05	dominated
$LPC_2^R+M^{15}$	1.62	264.87	dominated
$LPC_2^R+M^{20}$	1.01	306.34	dominated
LPC_2^R+3O	3.7	271.11	dominated

Table 5: Dominated and non-dominated algorithmic variants proposed by Staněk et al. (2019) for the AngleTSP instances.

Algorithmic variant	$\overline{\Delta}_{BKS}(\%)$	$\bar{t}(s)$	dominated/not dominated
CIF	12.62	3.49	dominated
$CIF+2O$	5.61	3.79	not dominated
$CIF+L$	12.13	40.48	dominated
$CIF+M^{15}$	1.72	25.13	not dominated
$CIF+M^{20}$	1.22	42.0	not dominated
$CIF+3O$	3.29	67.35	dominated
NN_S	13.28	147.56	dominated
NN_S+2O	5.66	147.89	dominated
NN_S+L	12.07	187.5	dominated
NN_S+M^{15}	1.45	175.55	dominated
NN_S+M^{20}	1.03	195.07	dominated
NN_S+3O	3.29	213.71	dominated
NN_S^L	12.46	241.49	dominated
NN_S^L+2O	5.21	241.83	dominated
NN_S^L+L	11.19	281.43	dominated
$NN_S^L+M^{15}$	1.36	268.74	dominated
$NN_S^L+M^{20}$	0.94	288.29	dominated
NN_S^L+3O	3.03	307.01	dominated
NN_S^2	2.34	69.36	dominated
NN_S^2+2O	2.34	69.42	dominated
NN_S^2+L	2.13	107.61	dominated
$NN_S^2+M^{15}$	0.64	90.97	not dominated
$NN_S^2+M^{20}$	0.49	106.27	not dominated
NN_S^2+3O	1.66	120.77	dominated
CH	52.32	0.74	dominated
$CH+2O$	8.41	1.3	dominated
$CH+L$	42.62	40.84	dominated
$CH+M^{15}$	2.29	39.76	dominated
$CH+M^{20}$	1.5	65.52	dominated
$CH+3O$	4.22	74.77	dominated
CH_C	47.85	0.9	dominated
CH_C+2O	7.99	1.44	dominated
CH_C+L	39.24	40.89	dominated
CH_C+M^{15}	2.26	38.58	dominated
CH_C+M^{20}	1.51	63.96	dominated
CH_C+3O	4.26	74.54	dominated
CH_C^L	17.86	0.56	not dominated
CH_C^L+2O	7.89	0.95	not dominated
CH_C^L+L	17.21	33.7	dominated
$CH_C^L+M^{15}$	1.74	30.5	dominated
$CH_C^L+M^{20}$	1.23	51.05	dominated
CH_C^L+3O	4.34	68.42	dominated
LPP	5.89	79.29	dominated
$LPP+2O$	2.65	79.52	dominated
$LPP+L$	5.3	117.33	dominated
$LPP+M^{15}$	0.76	99.76	dominated
$LPP+M^{20}$	0.59	115.46	dominated
$LPP+3O$	1.57	132.85	dominated
LPP^R	6.45	145.14	dominated
LPP^R+2O	2.45	145.38	dominated
LPP^R+L	5.42	184.4	dominated
LPP^R+M^{15}	0.7	166.58	dominated
LPP^R+M^{20}	0.46	182.29	dominated
LPP^R+3O	1.37	197.83	dominated
LPC_1^R	5.62	144.7	dominated
LPC_1^R+2O	2.37	144.94	dominated
LPC_1^R+L	4.78	183.76	dominated
$LPC_1^R+M^{15}$	0.64	165.91	dominated
$LPC_1^R+M^{20}$	0.42	180.77	not dominated
LPC_1^R+3O	1.33	197.69	dominated
LPC_2^R	5.25	154.52	dominated
LPC_2^R+2O	2.27	154.75	dominated
LPC_2^R+L	4.49	193.57	dominated
$LPC_2^R+M^{15}$	0.61	175.7	dominated
$LPC_2^R+M^{20}$	0.4	190.72	not dominated
LPC_2^R+3O	1.27	206.42	dominated

Table 6: Dominated and non-dominated algorithmic variants proposed by Staněk et al. (2019) for the AngleDistanceTSP instances.

B GTS-angular detailed results

Tables 7 and 8 report the comparison of the results grouped by instance size of the non dominated algorithms of Staněk et al. (2019) with GTS-angular r-LENS for the AngleTSP and AngleDistanceTSP instances, respectively. Tables 9 and 10 report the comparison of the results grouped by instance size of the non dominated algorithms of Staněk et al. (2019) with GTS-angular c-LENS for the AngleTSP and AngleDistanceTSP instances, respectively.

	$LPP^R + M^{15}$		$LPP^R + M^{20}$		$LPC_1^R + M^{15}$		$LPC_1^R + M^{20}$		GTS-angular r-LENS		
Size	$\overline{\Delta}_{BKS}(\%)$	$\bar{t}(s)$	$\overline{\Delta}_{BKS}(\%)$	$\bar{t}(s)$	$\overline{\Delta}_{BKS}(\%)$	$\bar{t}(s)$	$\overline{\Delta}_{BKS}(\%)$	$\bar{t}(s)$	$\overline{\Delta}^{best}_{BKS}(\%)$	$\overline{\Delta}^{avg}_{BKS}(\%)$	$\bar{t}^{avg}(s)$
5	0.0	0.01	0.0	0.01	0.0	0.01	0.0	0.01	0.0	0.0	0.03
10	0.0	0.06	0.0	0.06	0.0	0.07	0.0	0.07	0.0	0.0	0.09
15	0.0	0.28	0.0	0.29	0.0	0.31	0.0	0.3	0.0	0.03	0.63
20	0.0	5.46	0.0	1.69	0.0	5.41	0.0	1.65	0.0	0.11	1.44
25	0.03	5.58	0.0	9.56	0.03	5.7	0.0	9.76	0.0	0.28	2.68
30	2.46	10.03	0.29	24.53	2.44	11.01	0.29	24.9	0.0	0.09	3.91
35	1.75	7.41	1.17	13.98	1.75	7.17	0.77	13.37	0.0	0.29	5.21
40	0.52	10.24	0.39	23.04	0.55	10.53	0.39	25.57	0.11	0.6	7.74
45	1.33	20.68	0.79	22.54	1.57	21.29	1.13	21.57	0.24	0.51	10.24
50	2.81	27.38	3.03	23.99	2.94	27.25	3.07	25.78	0.12	0.48	12.63
55	1.15	27.99	1.4	26.96	1.56	26.71	1.42	24.92	0.18	0.86	15.87
60	2.27	36.28	1.1	71.85	2.14	36.27	1.12	59.03	0.51	1.3	21.91
65	2.0	41.28	1.37	54.37	2.03	39.05	1.21	54.72	0.07	0.8	21.97
70	3.54	41.59	2.49	60.89	3.26	38.7	2.15	71.02	0.3	1.2	29.4
75	3.68	50.6	3.16	80.5	2.91	47.68	2.22	74.42	0.52	1.62	37.17
80	1.99	60.12	1.52	89.3	2.06	57.13	1.45	83.75	-0.42	1.03	46.39
85	1.89	83.16	0.94	108.56	1.77	85.76	1.74	110.56	-0.47	0.64	52.83
90	1.84	82.47	1.13	98.81	2.04	86.07	1.47	113.28	-0.44	0.59	65.1
95	2.65	113.82	1.4	122.74	2.3	111.52	0.9	152.69	-0.28	1.29	83.41
100	1.78	105.39	1.85	128.51	0.96	126.97	1.24	137.37	-1.25	0.34	93.25
105	1.49	130.38	0.73	153.83	1.29	135.38	0.98	147.8	-1.32	-0.11	110.56
110	1.23	136.95	0.78	193.16	1.08	144.27	1.37	197.11	-1.14	0.48	135.13
115	3.12	152.72	2.05	219.16	2.13	160.4	1.06	220.23	-1.42	1.32	159.72
120	1.69	193.52	2.0	222.77	2.51	206.9	0.86	277.73	-0.42	1.33	165.45
125	1.65	197.23	0.82	278.65	1.57	199.95	1.19	247.45	-1.2	1.06	208.09
130	2.04	242.39	0.78	318.37	1.08	263.51	0.71	322.87	-1.27	1.48	230.43
135	2.61	260.75	0.74	339.41	1.31	264.54	1.45	302.15	-1.14	0.75	262.43
140	2.35	307.75	1.33	369.05	1.81	346.23	0.9	526.96	-1.02	1.09	309.86
145	2.74	335.4	1.69	431.07	1.9	348.53	0.77	414.82	-0.63	2.63	349.39
150	3.13	392.3	2.03	441.49	1.77	398.37	0.26	434.51	-0.91	1.6	425.24
155	1.86	408.28	1.27	454.77	1.71	415.5	1.1	458.93	-0.22	1.93	500.72
160	1.62	473.39	0.96	530.59	2.01	474.35	1.11	532.3	-0.69	1.78	481.71
165	2.12	515.59	1.36	548.05	2.02	507.8	1.1	546.1	-0.36	2.74	583.21
170	2.35	546.74	1.46	616.33	0.69	569.27	0.5	591.46	-0.85	1.98	692.26
175	1.67	615.93	1.28	671.89	1.5	649.39	0.77	681.89	-0.8	2.54	750.81
180	2.16	639.05	1.0	679.14	1.14	677.77	0.84	738.75	0.4	3.61	777.14
185	2.56	759.18	1.39	883.39	2.1	790.37	0.96	892.27	0.47	3.91	948.39
190	3.1	815.31	1.44	952.49	1.58	879.91	1.24	1012.42	0.02	3.43	1065.49
195	2.73	942.87	1.92	1190.87	2.15	1010.92	0.97	1243.17	-0.76	2.48	1091.55
200	1.96	1042.6	1.04	1153.63	1.6	1064.94	0.51	1187.44	-0.4	3.27	1271.36
Avg per inst	1.9%	245.95	1.2%	290.26	1.58%	256.32	0.98%	299.53	-0.36%	1.28%	275.77

Table 7: AngleTSP results: comparison of the results averaged by instance size of the non-dominated algorithms of Staněk et al. (2019) with GTS-angular r-LENS.

	$CIF+M^{15}$		$CIF+M^{20}$		$NN_S^2+M^{15}$		$NN_S^2+M^{20}$		$LPC_1^R+M^{20}$		$LPC_2^R+M^{20}$		GTS-angular r-LENS		
Size	$\overline{\Delta}_{BKS}(\%)$	$\bar{t}(s)$	$\overline{\Delta}_{BKS}(\%)$	$\bar{t}(s)$	$\overline{\Delta}_{BKS}(\%)$	$\bar{t}(s)$	$\overline{\Delta}_{BKS}(\%)$	$\bar{t}(s)$	$\overline{\Delta}_{BKS}(\%)$	$\bar{t}(s)$	$\overline{\Delta}_{BKS}(\%)$	$\bar{t}(s)$	$\overline{\Delta}^{best}_{BKS}(\%)$	$\overline{\Delta}^{avg}_{BKS}(\%)$	$\bar{t}^{avg}(s)$
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.01	0.0	0.01	0.0	0.0	0.06
10	0.0	0.04	0.0	0.04	0.0	0.04	0.0	0.04	0.0	0.06	0.0	0.06	0.0	0.0	0.25
15	0.0	0.12	0.0	0.12	0.0	0.14	0.0	0.15	0.0	0.2	0.0	0.19	0.0	1.82	0.6
20	0.0	1.38	0.0	0.36	0.0	1.45	0.0	0.43	0.0	0.55	0.0	0.55	0.06	1.49	1.13
25	0.18	2.05	0.0	2.77	0.0	2.26	0.0	3.05	0.0	3.27	0.0	3.3	0.0	0.7	2.02
30	0.47	2.11	0.31	4.06	0.0	2.17	0.0	4.38	0.25	5.5	0.25	5.6	0.0	0.45	3.1
35	0.89	2.52	0.37	4.7	0.05	2.84	0.0	4.89	0.05	5.76	0.05	5.92	0.0	0.24	4.51
40	1.05	2.6	0.21	5.0	0.06	3.4	0.06	6.37	0.0	7.89	0.0	8.21	0.0	0.18	6.15
45	0.6	6.61	0.32	6.05	0.0	7.57	0.0	6.93	0.12	9.93	0.1	10.25	0.0	0.15	8.27
50	1.26	8.02	0.84	9.13	0.17	8.66	0.05	9.28	0.29	12.59	0.29	13.15	0.0	0.1	10.54
55	1.19	7.89	1.47	7.73	0.39	10.48	0.37	8.64	0.5	12.85	0.44	13.24	0.0	0.15	13.67
60	0.65	9.0	0.34	17.18	0.52	11.32	0.59	18.69	0.11	24.81	0.11	25.55	0.0	0.12	16.81
65	1.35	12.22	0.89	20.12	0.25	14.46	0.25	24.78	0.83	30.19	0.53	29.73	0.0	0.05	20.15
70	1.35	10.09	1.83	21.88	0.2	15.83	0.25	24.63	0.63	38.7	0.56	37.66	0.0	0.09	25.29
75	1.82	10.2	1.07	26.36	0.65	19.66	0.42	35.0	0.45	45.07	0.55	45.86	0.0	0.17	33.22
80	2.36	11.81	1.49	22.93	0.56	21.6	0.39	30.56	0.49	43.65	0.47	45.73	0.0	0.2	38.54
85	2.15	15.49	1.77	26.93	0.52	30.23	0.49	36.46	0.38	55.55	0.36	57.8	0.01	0.15	43.53
90	1.83	20.94	1.48	20.17	0.79	34.63	0.68	40.85	1.33	66.87	1.36	70.15	0.0	0.13	50.74
95	1.82	25.65	1.72	25.4	0.61	45.96	0.47	44.04	1.0	77.18	0.88	81.4	0.0	0.13	61.93
100	1.35	18.67	1.48	30.48	0.64	43.03	0.48	50.12	0.69	80.62	0.55	85.43	0.0	0.2	76.2
105	2.21	22.93	1.89	24.95	0.47	50.64	0.27	62.88	0.52	87.43	0.52	92.15	-0.14	0.02	85.49
110	2.36	19.93	2.1	41.49	0.56	57.12	0.46	76.19	0.75	115.24	0.57	122.17	-0.11	0.03	103.88
115	2.59	22.01	1.65	42.88	1.05	64.2	1.09	83.4	0.56	133.99	0.59	141.12	-0.32	-0.13	119.66
120	1.61	24.21	1.41	66.45	0.8	72.24	0.65	89.68	0.44	141.49	0.43	149.73	-0.2	-0.03	128.14
125	1.51	25.05	0.99	52.87	1.01	78.11	0.88	101.62	0.62	170.8	0.64	181.08	-0.42	-0.15	151.93
130	2.44	28.91	1.61	63.75	1.06	99.16	0.69	130.14	0.65	202.11	0.57	210.07	-0.4	-0.19	156.87
135	1.77	34.0	0.82	61.18	0.79	100.42	0.4	127.25	0.5	210.86	0.51	226.86	-0.36	-0.1	198.43
140	2.9	44.19	1.41	70.69	0.81	118.86	0.39	148.59	0.65	217.13	0.56	228.75	-0.24	0.05	230.41
145	2.55	37.22	1.86	60.59	1.27	125.52	1.0	153.48	0.24	241.73	0.25	252.62	-0.28	0.12	257.08
150	2.26	45.73	1.25	75.74	0.89	142.58	0.68	161.55	0.27	275.2	0.3	292.31	-0.61	-0.24	273.42
155	2.92	44.56	1.86	69.11	0.75	159.1	0.69	177.47	0.25	307.26	0.31	326.27	-0.44	-0.1	331.01
160	2.27	60.29	1.83	80.66	1.03	195.69	1.0	195.61	0.74	342.48	0.6	360.37	-0.39	-0.05	355.7
165	2.78	47.32	1.86	70.03	1.58	189.79	0.97	209.97	0.31	372.3	0.29	389.28	-0.41	0.03	406.09
170	2.13	48.64	1.97	66.7	1.16	214.52	0.71	223.97	0.25	436.59	0.25	460.11	-0.43	-0.03	433.99
175	2.29	47.67	1.78	72.94	1.3	222.9	0.95	248.05	0.29	383.63	0.23	406.33	-0.5	-0.01	488.47
180	2.53	53.58	1.62	69.9	1.4	247.68	1.15	273.46	0.5	460.18	0.52	488.65	-0.65	-0.27	538.34
185	3.1	56.37	2.12	102.07	0.92	272.33	0.64	315.75	0.5	565.75	0.5	597.65	-0.65	-0.14	574.34
190	2.49	56.87	1.23	106.61	1.16	288.47	0.8	339.11	0.27	608.06	0.27	643.19	-0.61	-0.18	699.51
195	2.97	55.46	2.19	108.61	1.06	322.36	1.1	372.38	0.77	702.52	0.78	744.02	-0.53	-0.11	751.6
200	2.79	63.03	1.7	121.29	1.08	341.45	0.62	410.93	0.78	734.85	0.78	776.33	-0.51	-0.11	826.26
Avg per inst	1.72%	25.13	1.22%	42.0	0.64%	90.97	0.49%	106.27	0.42%	180.77	0.4%	190.72	-0.2%	0.12%	188.18

Table 8: AngleDistanceTSP results: comparison of the results averaged by instance size of the non-dominated algorithms of Staněk et al. (2019) with GTS-angular r-LENS.

	$LPP^R + M^{15}$		$LPP^R + M^{20}$		$LPC_1^R + M^{15}$		$LPC_1^R + M^{20}$		GTS-angular c-LENS		
Size	$\overline{\Delta}_{BKS}(\%)$	$\bar{t}(s)$	$\overline{\Delta}_{BKS}(\%)$	$\bar{t}(s)$	$\overline{\Delta}_{BKS}(\%)$	$\bar{t}(s)$	$\overline{\Delta}_{BKS}(\%)$	$\bar{t}(s)$	$\overline{\Delta}^{best}_{BKS}(\%)$	$\overline{\Delta}^{avg}_{BKS}(\%)$	$\bar{t}^{avg}(s)$
5	0.0	0.01	0.0	0.01	0.0	0.01	0.0	0.01	0.0	0.0	0.06
10	0.0	0.06	0.0	0.06	0.0	0.07	0.0	0.07	0.0	0.0	0.16
15	0.0	0.28	0.0	0.29	0.0	0.31	0.0	0.3	0.0	0.0	0.74
20	0.0	5.46	0.0	1.69	0.0	5.41	0.0	1.65	0.0	0.04	1.7
25	0.03	5.58	0.0	9.56	0.03	5.7	0.0	9.76	0.06	0.29	3.03
30	2.46	10.03	0.29	24.53	2.44	11.01	0.29	24.9	0.0	0.13	4.2
35	1.75	7.41	1.17	13.98	1.75	7.17	0.77	13.37	0.13	0.3	6.12
40	0.52	10.24	0.39	23.04	0.55	10.53	0.39	25.57	0.08	0.49	8.67
45	1.33	20.68	0.79	22.54	1.57	21.29	1.13	21.57	0.1	0.45	11.9
50	2.81	27.38	3.03	23.99	2.94	27.25	3.07	25.78	0.06	0.46	15.02
55	1.15	27.99	1.4	26.96	1.56	26.71	1.42	24.92	0.16	0.85	19.64
60	2.27	36.28	1.1	71.85	2.14	36.27	1.12	59.03	0.4	1.27	24.28
65	2.0	41.28	1.37	54.37	2.03	39.05	1.21	54.72	0.12	0.93	27.3
70	3.54	41.59	2.49	60.89	3.26	38.7	2.15	71.02	0.25	0.86	35.49
75	3.68	50.6	3.16	80.5	2.91	47.68	2.22	74.42	0.59	1.51	46.15
80	1.99	60.12	1.52	89.3	2.06	57.13	1.45	83.75	-0.42	1.02	55.7
85	1.89	83.16	0.94	108.56	1.77	85.76	1.74	110.56	-0.17	0.62	62.92
90	1.84	82.47	1.13	98.81	2.04	86.07	1.47	113.28	-0.46	0.79	79.93
95	2.65	113.82	1.4	122.74	2.3	111.52	0.9	152.69	0.28	1.24	105.76
100	1.78	105.39	1.85	128.51	0.96	126.97	1.24	137.37	-1.11	0.06	114.58
105	1.49	130.38	0.73	153.83	1.29	135.38	0.98	147.8	-1.36	-0.16	148.03
110	1.23	136.95	0.78	193.16	1.08	144.27	1.37	197.11	-0.88	0.33	162.97
115	3.12	152.72	2.05	219.16	2.13	160.4	1.06	220.23	-0.58	1.0	205.76
120	1.69	193.52	2.0	222.77	2.51	206.9	0.86	277.73	-0.22	1.41	200.73
125	1.65	197.23	0.82	278.65	1.57	199.95	1.19	247.45	-0.77	1.08	232.04
130	2.04	242.39	0.78	318.37	1.08	263.51	0.71	322.87	-0.41	1.83	281.73
135	2.61	260.75	0.74	339.41	1.31	264.54	1.45	302.15	-0.88	0.95	290.01
140	2.35	307.75	1.33	369.05	1.81	346.23	0.9	526.96	-0.2	1.92	342.55
145	2.74	335.4	1.69	431.07	1.9	348.53	0.77	414.82	0.58	2.39	419.58
150	3.13	392.3	2.03	441.49	1.77	398.37	0.26	434.51	-0.73	1.32	504.84
155	1.86	408.28	1.27	454.77	1.71	415.5	1.1	458.93	-0.23	1.45	608.08
160	1.62	473.39	0.96	530.59	2.01	474.35	1.11	532.3	-0.72	1.68	621.5
165	2.12	515.59	1.36	548.05	2.02	507.8	1.1	546.1	-0.58	1.94	706.46
170	2.35	546.74	1.46	616.33	0.69	569.27	0.5	591.46	-0.07	2.94	737.17
175	1.67	615.93	1.28	671.89	1.5	649.39	0.77	681.89	-0.14	2.57	889.16
180	2.16	639.05	1.0	679.14	1.14	677.77	0.84	738.75	-0.07	3.07	1031.58
185	2.56	759.18	1.39	883.39	2.1	790.37	0.96	892.27	1.1	4.22	1172.78
190	3.1	815.31	1.44	952.49	1.58	879.91	1.24	1012.42	0.75	3.82	1304.23
195	2.73	942.87	1.92	1190.87	2.15	1010.92	0.97	1243.17	0.3	3.43	1214.52
200	1.96	1042.6	1.04	1153.63	1.6	1064.94	0.51	1187.44	-0.09	3.07	1627.76
Avg per inst	1.9%	245.95	1.2%	290.26	1.58%	256.32	0.98%	299.53	-0.13%	1.29%	333.12

Table 9: AngleTSP results: comparison of the the results averaged by instance size of the non-dominated algorithms of Staněk et al. (2019) with GTS-angular c-LENS.

	$CIF+M^{15}$		$CIF+M^{20}$		$NN_S^2+M^{15}$		$NN_S^2+M^{20}$		$LPC_1^R+M^{20}$		$LPC_2^R+M^{20}$		GTS-angular c-LENS		
Size	$\overline{\Delta}_{BKS}(\%)$	$\bar{t}(s)$	$\overline{\Delta}_{BKS}(\%)$	$\bar{t}(s)$	$\overline{\Delta}_{BKS}(\%)$	$\bar{t}(s)$	$\overline{\Delta}_{BKS}(\%)$	$\bar{t}(s)$	$\overline{\Delta}_{BKS}(\%)$	$\bar{t}(s)$	$\overline{\Delta}_{BKS}(\%)$	$\bar{t}(s)$	$\overline{\Delta}^{best}_{BKS}(\%)$	$\overline{\Delta}^{avg}_{BKS}(\%)$	$\bar{t}^{avg}(s)$
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.01	0.0	0.01	0.0	0.0	0.07
10	0.0	0.04	0.0	0.04	0.0	0.04	0.0	0.04	0.0	0.06	0.0	0.06	0.0	0.0	0.3
15	0.0	0.12	0.0	0.12	0.0	0.14	0.0	0.15	0.0	0.2	0.0	0.19	0.0	0.61	0.68
20	0.0	1.38	0.0	0.36	0.0	1.45	0.0	0.43	0.0	0.55	0.0	0.55	0.0	0.58	1.32
25	0.18	2.05	0.0	2.77	0.0	2.26	0.0	3.05	0.0	3.27	0.0	3.3	0.0	0.53	2.23
30	0.47	2.11	0.31	4.06	0.0	2.17	0.0	4.38	0.25	5.5	0.25	5.6	0.0	0.12	3.35
35	0.89	2.52	0.37	4.7	0.05	2.84	0.0	4.89	0.05	5.76	0.05	5.92	0.0	0.15	4.89
40	1.05	2.6	0.21	5.0	0.06	3.4	0.06	6.37	0.0	7.89	0.0	8.21	0.0	0.25	6.15
45	0.6	6.61	0.32	6.05	0.0	7.57	0.0	6.93	0.12	9.93	0.1	10.25	0.0	0.23	8.4
50	1.26	8.02	0.84	9.13	0.17	8.66	0.05	9.28	0.29	12.59	0.29	13.15	0.0	0.21	10.46
55	1.19	7.89	1.47	7.73	0.39	10.48	0.37	8.64	0.5	12.85	0.44	13.24	0.0	0.2	13.24
60	0.65	9.0	0.34	17.18	0.52	11.32	0.59	18.69	0.11	24.81	0.11	25.55	0.0	0.14	16.82
65	1.35	12.22	0.89	20.12	0.25	14.46	0.25	24.78	0.83	30.19	0.53	29.73	0.0	0.07	20.58
70	1.35	10.09	1.83	21.88	0.2	15.83	0.25	24.63	0.63	38.7	0.56	37.66	0.01	0.12	26.16
75	1.82	10.2	1.07	26.36	0.65	19.66	0.42	35.0	0.45	45.07	0.55	45.86	0.02	0.26	29.9
80	2.36	11.81	1.49	22.93	0.56	21.6	0.39	30.56	0.49	43.65	0.47	45.73	0.04	0.15	39.23
85	2.15	15.49	1.77	26.93	0.52	30.23	0.49	36.46	0.38	55.55	0.36	57.8	0.03	0.17	43.58
90	1.83	20.94	1.48	20.17	0.79	34.63	0.68	40.85	1.33	66.87	1.36	70.15	0.03	0.28	52.3
95	1.82	25.65	1.72	25.4	0.61	45.96	0.47	44.04	1.0	77.18	0.88	81.4	0.06	0.24	60.78
100	1.35	18.67	1.48	30.48	0.64	43.03	0.48	50.12	0.69	80.62	0.55	85.43	0.06	0.29	62.8
105	2.21	22.93	1.89	24.95	0.47	50.64	0.27	62.88	0.52	87.43	0.52	92.15	-0.09	0.16	82.44
110	2.36	19.93	2.1	41.49	0.56	57.12	0.46	76.19	0.75	115.24	0.57	122.17	-0.06	0.17	96.33
115	2.59	22.01	1.65	42.88	1.05	64.2	1.09	83.4	0.56	133.99	0.59	141.12	-0.23	0.02	111.14
120	1.61	24.21	1.41	66.45	0.8	72.24	0.65	89.68	0.44	141.49	0.43	149.73	-0.17	-0.05	117.91
125	1.51	25.05	0.99	52.87	1.01	78.11	0.88	101.62	0.62	170.8	0.64	181.08	-0.28	-0.03	150.64
130	2.44	28.91	1.61	63.75	1.06	99.16	0.69	130.14	0.65	202.11	0.57	210.07	-0.24	0.09	143.72
135	1.77	34.0	0.82	61.18	0.79	100.42	0.4	127.25	0.5	210.86	0.51	226.86	-0.28	0.11	162.28
140	2.9	44.19	1.41	70.69	0.81	118.86	0.39	148.59	0.65	217.13	0.56	228.75	-0.15	0.11	194.56
145	2.55	37.22	1.86	60.59	1.27	125.52	1.0	153.48	0.24	241.73	0.25	252.62	-0.12	0.3	213.16
150	2.26	45.73	1.25	75.74	0.89	142.58	0.68	161.55	0.27	275.2	0.3	292.31	-0.44	0.02	254.19
155	2.92	44.56	1.86	69.11	0.75	159.1	0.69	177.47	0.25	307.26	0.31	326.27	-0.37	0.01	288.69
160	2.27	60.29	1.83	80.66	1.03	195.69	1.0	195.61	0.74	342.48	0.6	360.37	-0.27	0.14	315.59
165	2.78	47.32	1.86	70.03	1.58	189.79	0.97	209.97	0.31	372.3	0.29	389.28	-0.22	0.21	357.41
170	2.13	48.64	1.97	66.7	1.16	214.52	0.71	223.97	0.25	436.59	0.25	460.11	-0.33	0.06	404.39
175	2.29	47.67	1.78	72.94	1.3	222.9	0.95	248.05	0.29	383.63	0.23	406.33	-0.19	0.19	464.94
180	2.53	53.58	1.62	69.9	1.4	247.68	1.15	273.46	0.5	460.18	0.52	488.65	-0.39	0.18	422.32
185	3.1	56.37	2.12	102.07	0.92	272.33	0.64	315.75	0.5	565.75	0.5	597.65	-0.27	0.12	488.78
190	2.49	56.87	1.23	106.61	1.16	288.47	0.8	339.11	0.27	608.06	0.27	643.19	-0.29	0.12	584.58
195	2.97	55.46	2.19	108.61	1.06	322.36	1.1	372.38	0.77	702.52	0.78	744.02	-0.12	0.31	647.09
200	2.79	63.03	1.7	121.29	1.08	341.45	0.62	410.93	0.78	734.85	0.78	776.33	-0.19	0.39	709.74
Avg per inst	1.72%	25.13	1.22%	42.0	0.64%	90.97	0.49%	106.27	0.42%	180.77	0.4%	190.72	-0.11%	0.18%	165.33

Table 10: AngleDistanceTSP results: comparison of the results averaged by instance size of the non-dominated algorithms of Staněk et al. (2019) with GTS-angular c-LENS.