Practical Methods for Computing Large Covering Tours and Cycle Covers with Turn Cost

Sándor P. Fekete* Dominik Krupke*

Abstract

We study the problem of computing provably optimal and near-optimal solutions for the NP-hard problem of finding covering tours and cycle covers with turn cost, which are of practical importance for a variety of applications, such as pest control and precision farming. Previous work has largely focused on theoretical aspects, such as complexity and approximation. We develop a number of algorithm engineering techniques and refinements to make such theoretical insights practically useful, resulting in a comprehensive study for solving a wide spectrum of large instances. We compute provably optimal solutions for instances with more than 1000 pixels, from the largest previous solved instance size of 76 (de Assis and de Souza 2011). Making use of additional algorithm engineering techniques for handling very large instances, we also compute near-optimal solutions for instances with up to 300 000 pixels, for which we give solutions that are typically within a few percent of our computed lower bounds. We also provide an experimental comparison of a practically refined version of our new theoretical approach with the approximation technique of Arkin et al. that dates back to 2001; we show that our new LP/IP-based approximation method closes 70% of the remaining optimality gap to the lower bound.

1 Introduction

The Traveling Salesman Problem (TSP) is one of the classic tasks of combinatorial optimization. Easy to describe but NP-hard to solve, it has given rise to a large body of research focusing on theoretical aspects such as complexity and approximation. On the practical side, the TSP has also stimulated significant work in algorithm engineering. Ever since Dantzig, Fulkerson and Johnson [14] in 1954 presented the provably optimal solution of a 49-city instance based on an (integer) linear-programming (IP/LP) approach, IP/LP methods have been used on a wide spectrum of other optimization problems. With a variety of additional techniques, the frontiers of TSP instance sizes for which provably optimal solutions can be computed have been pushed all the way to 85,900 cities; see [8] for a comprehensive overview.

Research on TSP has also served as the blueprint for a wide range of other real-world problems, many of them motivated by generalizations or modifications, such as lawn mowing and milling, where visiting a discrete set of points is replaced by covering a geometric region with a tool or sensor. Other variants arise from modified objective functions or additional constraints, such as the total turn of a tour instead of its length.

Figure 1: Satellite image of a real-world instance arising from mechanized agriculture. Notice the tracks resulting from the use of large-scale agricultural machinery, and the uneven crop yield, motivating subset and penalty versions of the problem. (Image: Google, GeoBasis-DE/BKG ©2009)

In this paper we present a number of algorithm engineering techniques for computing covering tours and cycle covers with turn cost, which are of practical significance in areas such as pest control and precision farming. (See our related video [12] `https://www.youtube.com/watch?v=SFyOMDgdNao` for an animated illustration in the context of fighting mosquitoes.) This includes variants such as the subset and the penalty versions of the problem, in which the objective is to cover appropriate subsets of the given region. True to the spirit of algorithm engineering, the work presented in this paper refines and extends our previous theoretical work [18] that discusses complexity and approximation.

1.1 Related Work The problem of minimizing the necessary distance for covering a given region by a moving tool is known as the *Lawnmower Problem*; if

*Department of Computer Science, TU Braunschweig, 38106 Braunschweig, Germany, {s.fekete,d.krupke}@tu-bs.de

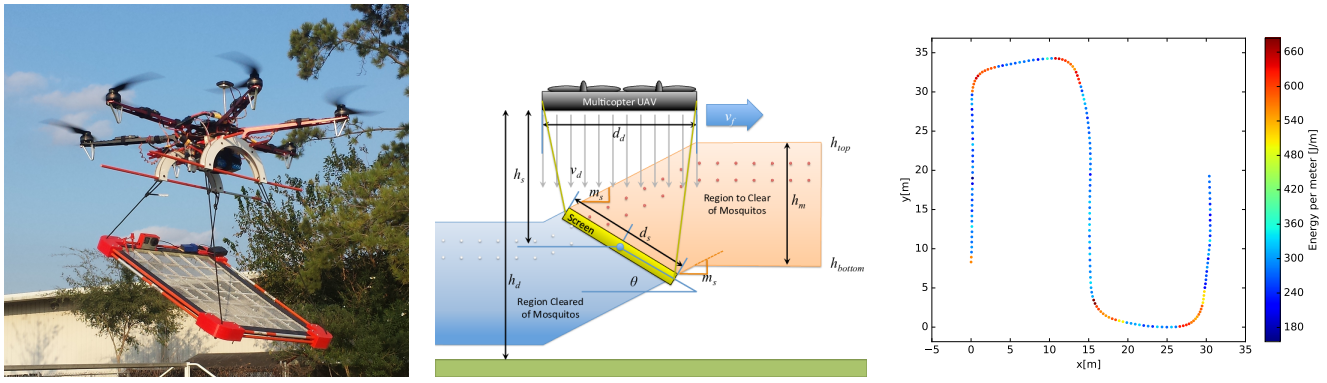

Figure 2: **(Left)** A drone equipped with an electrical grid for killing mosquitoes. **(Middle)** Physical aspects of the flying drone. **(Right)** Making turns is expensive. See our related video at `https://www.youtube.com/watch?v=SFyOMDgdNao` for details, and [12] for an accompanying abstract.

the tool is required to stay within the region, we are dealing with the *Milling Problem*. See Arkin et al. [11] for a theoretical study, providing approximation algorithms for both variants. They give a 2.5-approximation for minimum length milling in orthogonal (not necessarily integral) polygons with a unit square cutter. For simple orthogonal polygons, an 11/5-approximation is given and, in case we can reduce the problem to finding a covering tour in a grid graph, a 6/5-approximation algorithm is given (which improves a previous result of Ntafos [29] of a 4/3-approximation algorithm). For the mowing variant, a $3+\epsilon$-approximation is provided that internally uses a PTAS for the Euclidean TSP.

For covering a discrete set of points in the $\mathbb{R}^2$ plane for which only the turning angles are measured, the problem is called the *Angular Metric TSP*. For this problem, the cost function changes from a relatively straightforward sum of edge weights to more involved combinations of edges at the vertices, making approximation more challenging. Aggarwal et al. [4] provide an $O(\log n)$ approximation algorithm for cycle covers and tours that works even for distance costs and higher dimensions. Fekete and Woeginger [19] consider the problem of connecting a point set with a tour for which the angles between the two successive edges are constrained. Finding a curvature-constrained shortest *path* with obstacles has been shown to be NP-hard by Lazard et al. [26]. Without obstacles, the problem is known as the *Dubins path* [17] that can be computed efficiently. For different types of obstacles, Boissonnat and Lazard [13], Agarwal et al. [2] and Agarwal and Wang [3] provide polynomial-time algorithms or $1+\epsilon$ approximation algorithms, respectively. Takei et al. [32] consider the solution of the problem from a practical perspective. The *Dubins Traveling Salesman Problem* is considered by Le Ny et al.[30]

For covering a geometric region in the presence of turn cost, Arkin et al. [9, 10] provide a first approximation algorithm for tours and cycle covers in grid graphs with turn cost and show hardness of the tour variant. Most closely related to this submission is our recent theoretical work [18] that provides important theoretical progress, resolving *Problem 53* in *The Open Problems Project* [16] by proving that finding a cycle cover of minimum turn cost is NP-hard, even in the restricted case of grid graphs. As a consequence, we showed that all relevant problem variants are NP-hard. We also proved that finding a subset cycle cover of minimum turn cost is NP-hard, even in the restricted case of *thin* grid graphs, in which no induced 2×2 subgraph exists. This differs from the case of full coverage in thin grid graphs, which is known to be polynomially solvable [10]. We also provided a general IP/LP-based technique for obtaining constant-factor approximations for all problem variants; some details are described in Section 3. This approach includes the first approximation algorithms for subset cycle covers and tours, as well as for penalty cycle covers and tours; these are also valid for travel costs that are linear combinations of turn and distance costs.

Algorithm engineering for covering tours and cycle covers with turn cost has been more limited. Maurer [27] proves that cycle *partition* with turn cost in grid graphs can be solved in polynomial time. He also performed integer programming experiments on cycle cover and cycle partition. De Assis and de Souza [15] considered integer programming for tours but were only able to solve small instances with up to 76 vertices. We considered their test instances for verification but they showed no challenge for further evaluation.

The generalization of the Angular Metric TSP to generic graphs is called the Quadratic Traveling Salesman Problem where the cost of an edge can depend on the preceding edge. The solvable instances even for

the geometric angular metric variant are usually very small (below 100 points), see e.g., [24], [31], [6].

Covering a polygonal area by a tour is an important practical problem and hence a considerable amount of work can be found. An extensive survey on mostly heuristic techniques for robots is given by Galceran and Carreras [21]. Two common techniques are putting a grid on the area as we do, see, e.g., Zelinsky et al. [34] or Gabriely and Rimon [20], or partitioning the polygon into simple geometric areas (e.g. trapezoidal map) and use simple patterns to cover these areas, see e.g., Huang [23].

Optimizing cutter paths is considered by Yao et al. [33] who also mention the increase of costs due to turns. Planning tractor tours for crop harvesting operations including turn penalties is analyzed by Ali et al. [7]. Ahmadzadeh et al. [5] and Agarwal et al. [1] look at covering tours of UAVs for area observation with restrictions on the turn radii.

1.2 Our Contribution In this paper we bridge the gap between theory and practice of finding covering tours and cycle covers with turn cost, and show how to apply algorithm engineering techniques in combination with refined modeling in order to greatly extend the size of instances for which provably optimal and near-optimal solutions can be computed.

We provide the following main results.

- We describe an efficient implementation of an approximation technique for full/subset/penalty coverage with cycle covers or tours.
- We provide a refined integer programming formulation that can solve instances with over 1000 pixels using a modern integer programming solver, greatly extending the magnitude of solvable instances from the previously published size of 76.
- We present a comprehensive computational study of instances with up to 300 000 pixels, for which we give solutions that are typically within a few percent of the computed lower bounds.
- We also provide a practical comparison of our approximation algorithm with the approximation technique of Arkin et al. that dates back to 2001; we show that our new LP/IP-based approximation method closes 70% of the remaining optimality gap to the lower bound for a wide range of benchmark instances.

2 Preliminaries

Given a grid graph $G = (P, E)$, where P are unit-sized squares on the integral grid, also called *pixels*, and two pixel are joined by an edge $e \in E$ iff they are adjacent. In addition, we may be given a subset $S \subseteq P$ of pixels or a penalty function $\rho : P \to \mathbb{Q}_0^+$. We consider tours and cycle covers in grid graphs with three variations: *Full coverage*, where every pixel has to be covered, *subset coverage*, where at least a specific subset S has to be covered, and *penalty coverage*, where every uncovered pixel $p \in P$ involves a penalty $\rho(p)$. The objective function is an arbitrary but fixed non-negative linear combination of the number of pixel transitions and turns. For the penalty variant, the objective function also contains the sum of penalties for uncovered pixel. The number of turns is measured in 90° turns, which we also call *simple turns*. The cost of a u-turn (a turn of 180°) corresponds to two simple turns. A *cycle* is a closed sequence of at least two adjacent pixels. A pixel is covered by a cycle if it is in its pixel sequence. A *cycle cover* is a set of cycles that together cover all pixels, while a *tour* is a cycle cover that consists of a single cycle. See Fig. 3 for an example.

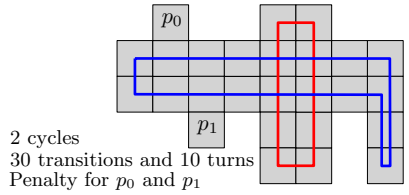

Figure 3: Example of a penalty cycle cover whose cost is a linear combination of the number of transitions and 90° turns and the penalties for not covering p_1 and p_2.

3 Efficient Implementation of an Approximation Algorithm

In [18] we describe an approximation technique that yields constant-factor approximations for full/subset/penalty cycle covers and tours in grid graphs (and even more generic geometric instances). The purpose of that proof was to establish a constant factor for the worst-case performance, not practical efficiency. As a consequence, the theoretical algorithm would struggle (especially regarding memory consumption) with instances larger than 10 000 pixels, even when exploiting duality properties to reduce the size of the auxiliary problems. In this section we give a number of additional algorithm engineering techniques to turn this worst-case performance into a significantly more efficient implementation, based on exploiting more refined properties. The result is an implementation that is able to solve instances with hundreds of thousands of pixels. The corresponding experimental evaluation is given in Sec. 5. For easier description, we initially focus

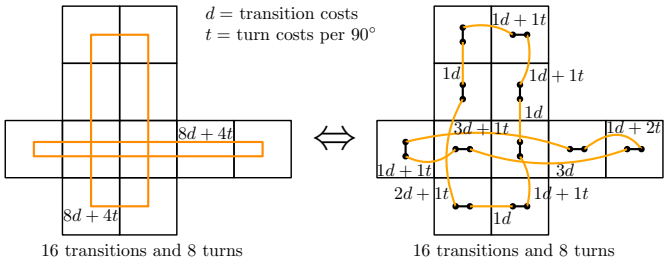

Figure 4: Finding the cheapest cycle cover and finding the ASC with the lowest perfect matching are equivalent problems. If we select the atomic strip that has either the correct entry or exit orientation, the cost of the matching edges equals the cost of the cycle. If the wrong atomic strip is selected, the cost can increase by two turns. The weight of the matching edges can be computed efficiently (in our efficient implementation, this is actually not necessary). Covering a pixel multiple times is not a problem, because the matching edges can skip pixels (for recreating the cycle cover, one just has to keep track of which ones).

on full coverage cycle covers. The necessary adaptations for the other variants are sketched in the end.

A fundamental part of the approximation technique are *atomic strips*, which allow us to push the turn cost into the edge weights. Similar to vertices in a grid graph, they encode *positions*; in addition each atomic strip also encodes an *orientation* by having two entries/exits in opposite directions. Hence, an atomic strip can be interpreted as a strip or line of zero (or infinitely small) length. For each pixel, we have a horizontal and a vertical atomic strip, of which at least one has to be in the solution to cover the pixel. Thus, an *Atomic Strip Cover* (ASC) is a selection of exactly one *atomic strip* per pixel. A cycle cover is obtained by computing a minimum weight perfect matching on the endpoints of the selected atomic strips. The weight of a matching edge equals the minimum cost of transiting between the two endpoints including the turns at the ends. It is straightforward to prove that the minimal matching over all possible ASCs corresponds to an optimal solution, see Fig. 4 and [18] for more theoretical details.

In the original algorithm we use an integer program (IP) that combines finding this ASC with its corresponding matching. This IP is then solved fractionally; for each pixel, the strip with the higher fractional value for an ASC is selected. We were able to show that the matching of this ASC yields a solution at most 4 times higher than the optimal solution by using polyhedral arguments. See Fig. 5 for an illustration.

We first show how to implement the ASC and matching parts of the approximation algorithm for (full) cycle cover. Afterwards we sketch how to obtain algorithms for subset and penalty coverage, as well as the tour variants. An extension to other grids is also possible.

3.1 Atomic Strip Covers Formulating the problem of finding the ASC with the best matching as an integer program requires $O(|P|)$ Boolean variables for the atomic strips and $O(|P|^2)$ Boolean variables for the matching edges. In addition, the weight of the matching edges has to be computed, which can be very time-consuming for large grid graphs.

There are multiple possibilities for formulating the cycle cover problem as an integer program. Almost all of these formulations are potential replacements for the linear relaxation of the original integer program, because the solutions can easily be transformed and the proof in [18] only requires a fractional solution that is a lower bound on the optimum.

LEMMA 3.1. ([18]) *A fractional solution for Atomic Strip Cover and matching (see Fig. 5 left) can be transformed into an integral solution of at most four times the objective value.*

We extract the Atomic Strip Cover from the fractional solution of the following IP, which we selected based on prior experiments. We use the non-negative variables $x_{ijk} = x_{kji} \in \mathbb{N}_0$ for pixel $p_j \in P$ and adjacent pixels $p_i, p_k \in N(p_j)$ that state how often the transition $p_i - p_j - p_k$ or $p_k - p_j - p_i$ is used, see Fig. 6. Let $\text{cost}_j(i,k) \in \mathbb{Q}_0^+$ map the cost of this transition. We minimize the overall coverage costs; Eq. (3.2) enforces a pixel to be covered and Eq. (3.3) enforces the transitions between two adjacent pixels to match.

$$\min \sum_{p_j \in P} \sum_{p_i, p_k \in N(p_j)} \text{cost}_j(i,k) \cdot x_{ijk} \tag{3.1}$$

$$\text{s.t.} \sum_{p_i, p_k \in N(p_j)} x_{ijk} \geq 1 \qquad \forall p_j \in P \tag{3.2}$$

$$2 \cdot x_{jij} + \sum_{p_k \in N(p_i), p_k \neq p_j} x_{jik} = 2 \cdot x_{iji} + \sum_{p_k \in N(p_j), p_k \neq p_i} x_{ijk} \qquad \forall \{p_i, p_j\} \in E \tag{3.3}$$

Obtaining a fractional Atomic Strip Cover from the fractional solution of this IP is done as follows. If a coverage variable represents a straight transition, add its value to the equally oriented strip, otherwise distribute the value equally between both, because both are valid selections. Our approximation algorithm selects the dominant strip, i.e., the strip with the highest value in

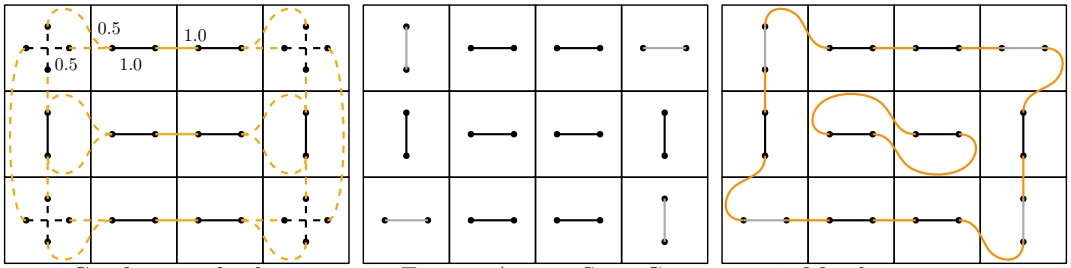

Figure 5: Example of the approximation algorithm for a simple full cycle cover in a grid graph. First a fractional solution for the atomic strip cover matching formulation is computed via linear programming. Strips and edges with value 0 are omitted, while dashed ones have a value of 0.5. Then the dominant (i.e., highest valued) atomic strips of this solution are selected. The grey atomic strips are ambiguous, i.e., we could have also chosen the other one. Finally, a minimum weight perfect matching on the ends of the atomic strips is computed. Recall that the atomic strips do not have any length (but only an orientation) so the curves in the corner are actually simple 90° turns.

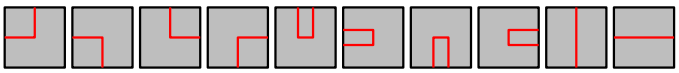

Figure 6: There are 10 ways of covering a pixel; each one corresponds to a variable of the IP.

the fractional solution. See Fig. 7 for an illustration. This Atomic Strip Cover equals the selection of an equal weighted fractional Atomic Strip Cover and matching as in the original algorithm in [18] and is hence a feasible replacement.

LEMMA 3.2. *A fractional solution of the IP* (3.1)-(3.3) *can be converted into a fractional solution (of equal objective value) of original IP, i.e., a fractional Atomic Strip Cover with a corresponding fractional matching.*

Proof. It is easy to see that for an integral cycle we can easily extract matching atomic strips and corresponding connecting edges of equal cost. For transforming a fractional solution, simply multiply the solution by some value z such that all cycles become integral. Do the transformation independently, i.e. do not care for double coverages, for every integral cycle and afterwards divide the solution again by z. Remove the superfluous coverages by connecting matching edges at both ends of an overused atomic strip directly which does not increase the cost but decreases the usage of the atomic strip. Note that the original integer program allowed loop edges in the fractional solution as they cannot appear in integral solutions and do not harm the proof.

3.2 Matching We now have an atomic strip cover, i.e., an atomic strip per pixel, that we now need to connect to a cycle cover via a matching on the end points. A naïve matching of the strips results in a quadratic size matching graph, where each edge represents a connection that has to be computed and stored. Even when utilizing the dual of the primal-dual matching algorithm to exclude edges, the memory and time consumption showed to be too high for larger instances ($> 10\,000$ pixels). However, one can use the simple structure of a grid graph to solve this problem much more efficiently: The matching in Fig. 4 can also be expressed as a matching that only uses short edges between two adjacent pixels if we are allowed to add additional strips, see Fig. 8. We can add such optional strips by having a zero-cost edge between its two endpoints, such that it disappears without cost if it is not needed. Recall that we match the end points of the atomic strips; thus, using the zero-cost edge effectively removes the corresponding atomic strip. Further, we can limit the maximally needed strips to two horizontal and two vertical strips due to the following lemma.

LEMMA 3.3. (ARKIN ET AL. [10]) *If a cycle cover covers a pixel more than four times or is passed straight more than two times in the same orientation, the length of the cycle cover and the coverage of the pixel can be reduced by local optimization without increasing the turns.*

The strip of the Atomic Strip Cover does not get such a removal edge, because it has to be used by the proper cycles, so we are left with 8^2 edges per two adjacent pixels and three zero-cost edges per pixel.

3.3 Subset and Penalty Coverage Subset coverage can be easily expressed as penalty coverage by making the penalty extremely high for mandatory pixel and zero for optional pixel. So we can limit ourselves to the penalty coverage variant. The idea for penalty coverage is to introduce an artificial cycle for each pixel with

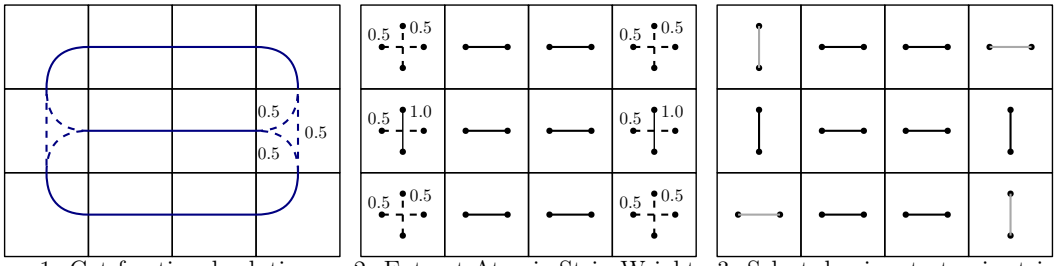

Figure 7: Obtaining the Atomic Strip Cover from the IP (3.1)-(3.3). Note that the overcoverage of some pixel is not a problem, because we can reduce it by connecting two matching edges of the endpoints of the atomic strip, i.e., skip it without increasing the cost.

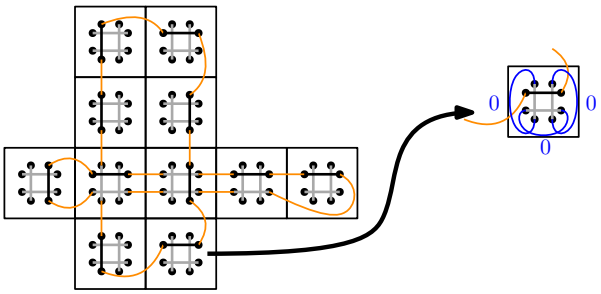

Figure 8: If we add optional atomic strips (gray), we only need matching edges between atomic strips of adjacent pixel. Optional atomic strips can be created by adding a zero-cost edge between its endpoints that allow it to be neutralized if not needed. We need at most three such optional atomic strips per pixel.

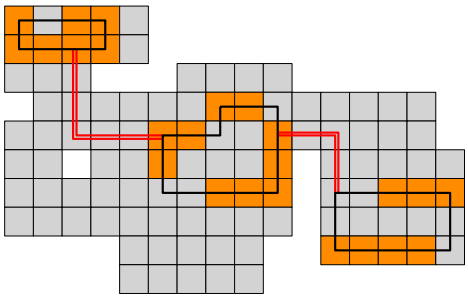

Figure 9: Connecting subset cycles (cycles in black, subset pixel in orange) by a minimum spanning tree (red edges) on the components/cycles.

the cost of the penalty that covers exactly this pixel. For this we can add an edge to each atomic strip that connects the both endpoints. The weight of such an edge is the penalty for not covering the corresponding cycle. When using IP (3.1)-(3.3) we only need to add an additional Boolean variable for each pixel that expresses using the corresponding penalty cycle. This variable needs to be added to the objective function and the coverage constraint. See [18] for more details on this idea.

3.4 Tours For full coverage one can always connect two adjacent cycles with at most two additional turns and two additional transitions as shown by Arkin et al. [10]. They also showed that if a pixel is covered more than four times by a tour, we can do a local optimization. Because every cycle has at least four turns and two transitions on its own, greedily connecting adjacent cycles provides us an approximation factor of 6.

For subset coverage we do not necessarily have adjacent cycles but one can build a graph where every cycle is a vertex and every edge is the cheapest connection between these two cycles. In [18] we show that connecting the cycles to a tour via a minimum spanning tree in this graph (see Fig. 9) provides us an approximation factor of 10.

For penalty coverage we also have to consider that we could decide not to cover pixels instead of connecting the cycles. In [18] we show that by using a 2-approximation for the Prize-Collecting Steiner Tree instead of a Minimum Spanning Tree, one can obtain a 12-approximation.

3.5 Other Grids The approximation technique including the just presented techniques for runtime improvement can also be applied to more generic grid graphs such as hexagonal or 3-dimensional grids. For hexagonal or 3-dimensional grids we would need three atomic strips per vertex. In general one has to make sure that for every possible coverage possibility of a vertex/pixel either incoming or the outgoing orientation matches the endpoint of an atomic strip. Of course this may need more optional atomic strips for the matching technique. Note that while we are using uniform distance costs here, the algorithm theoretically allows the usage of heterogeneous edge lengths. Also the turning angles would allow some heterogeneity and hence one could for example also compute solutions for 'warped'

grid graphs. Obtaining 'well deformed' grid graphs for natural instances, e.g. fields for harvesting, instead of just putting a uniform grid over it is a potential next step to improve the practicality of this approach.

4 Integer Programming

We provide further details on an integer programming formulation for solving the full coverage tour problem to optimality. The IP in Eq. (3.1) to (3.3) described in Sec. 3.1 already provides a powerful IP formulation for the cycle cover problem. We experimented with different formulations; this turned out to be the best for most cases and variants. It is analogous to the cycle cover formulation used by Maurer [27]; in order to make it useful for tours, it only lacks constraints for subtour elimination. These subtour elimination constraints are more complicated than for the classic TSP, because here cycles can intersect without being connected. We describe two different variants: a simple but insufficient one analogous to the one for TSP, and a more complex one that is sufficient.

Note that the more general penalty coverage variant is described in [28]. The difficulty with penalty tours is that when enforcing connectivity one has to consider the possibility of paying the penalty.

4.1 A Necessary Family of Simple Tour Constraints

Let C be a subtour that is not crossed by any other cycle. It is easy to see that there has to be a variable used that allows to leave the set of pixels $P' = \{p_i \in C\}$ covered by C.

$$\sum_{p_j \in P', i,k \in N(p_j), p_i \notin P'} x_{ijk} \geq 1 \quad \forall P' \subsetneq P, P' \neq \emptyset \tag{4.4}$$

This is valid for any real non-empty subset $P' \subsetneq P$ of pixels, independent of any cycle. Therefore, C may also consist of, e.g., two intersecting cycles, as long as C does not fully cover all pixel P. This constraint forces cycles to intersect but the turn costs prevent us from assuming that this is sufficient for connecting the cycles. For example the two cycles in Fig. 10 intersect and hence fulfill above constraint but connecting them would cost us two extra turns.

4.2 A Sufficient Family of Advanced Tour Constraints

The following constraint prohibits all subtours C that have a pixel p_f only covered by C and another pixel $p_{f'}$ not covered by C. For cycle covers that do not already contain a tour, you can argue the existence of such pixels by using the corner pixels which can only be covered by one cycle (as they automatically connect all their cycles). The constraint enforces that either p_f is passed differently, or there is a turn in a pixel traversed without a turn by C, or a currently unused variable is used in one of the other pixels of C. Let F_s be the set of pixels different from p_f that are traversed without a turn by C. Other cycles may also pass pixels in F_s, but not by a simple turn as otherwise they would be connected with C. Let $T(p_i)$ be the simple turn variables for the pixel $p_i \in P$ and let x' denote the value of a variable in the current solution.

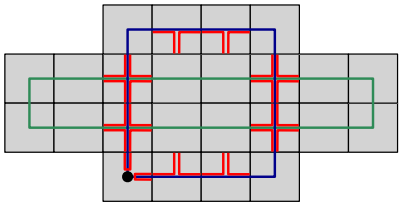

Figure 10: The blue and the green cycle intersect, so the first tour constraint is satisfied even though the two cycles are not connected. We can use the second tour constraint to force the cycles to connect using the black dotted pixel as p_f. The constraint forces at least one of the red traversals to be used.

$$\sum_{i,j \in N(p_f), x'_{ifj}=0} x_{ifj} + \sum_{t \in T(p_i), p_i \in F_s} t + \sum_{p_j \in C \setminus (F_s \cup \{p_f\}), i \neq k \in N(p_j), x'_{ijk}=0} x_{ijk} \geq 1 \tag{4.5}$$

An example of this constraint can be seen in Fig. 10. This constraint has to be constructed and added for every intermediate solution that is not a tour (but only a cycle cover).

5 Practical Computation

We implemented and tuned our approximation technique as well as an optimal solver based on integer programming for full, subset, and penalty cycle covers and tours in grid graphs. We used CPLEX 12.7.1.0 and the Minimum-Weight Perfect Matching implementation of Kolmogorov [25]. For connecting the penalty cycle cover to a tour, we used a Prize-Collection Steiner Tree implementation (proven 2-approximation) of Hedge, Indyk, and Schmidt [22]. Furthermore, we implemented the approximation algorithm of Arkin et al. as a comparison. The (C++)-code, test instance examples, and further material can be found on `https://github.com/d-krupke/turncost`.

We used two different generators for creating random test instances that are motivated by floor plans. The first one (Type I) uses a random orthogonal polygon with 20 to 200 vertices and selects all points of a grid inside of

Figure 11: An example instance of Type I with 7684 pixels.

it; see Fig. 11 for an example. The second one (Type II) takes a union of random rectangles, with the maximum size of a rectangle bounded by $O(\sqrt{N})$, N being the desired size. We used two different parameter settings to obtain instances of different granularity, denoted by Type IIa and Type IIb; examples are shown in Fig. 12 for Type IIa and Fig. 13 for Type IIb. While the test instance for Type II have a uniform size distribution, for Type I there are more smaller than larger instances, because the amount of pixel is harder to control in the generation process.

The practical computations were performed on regular desktop computers equipped with an *Intel(R) Core(TM) i7-6700K CPU @* 4.00 GHz and 64 GB of RAM. The used CPLEX version was 12.7.1.0 with the parameters `EpInt`=0, `EpGap`=0, `EpOpt`=1×10^{-9}, and `EpAGap`=0. For integer programming, CPLEX was supplied with solutions of the approximation algorithm. In all cases, the objective function models the travel cost arising from a linear combination of turn and distance cost, with a range of different coefficients, accounting for different relative cost of making turns.

In Fig. 15 we see how many instances of Type II can be solved to optimality within 15 min. One can see that instances of Type IIa seem to be harder than instances of Type IIb, which is surprising, because instances of Type IIb have the more complicated polygons. Further we evaluated the common and usually much more efficient way of adding the subtour elimination constraints directly to every integral solution in the branch-and-bound process via callbacks versus only adding these to optimal solutions and restart the solving process until a connected solution is obtained. Surprisingly the second variant is able to solve more

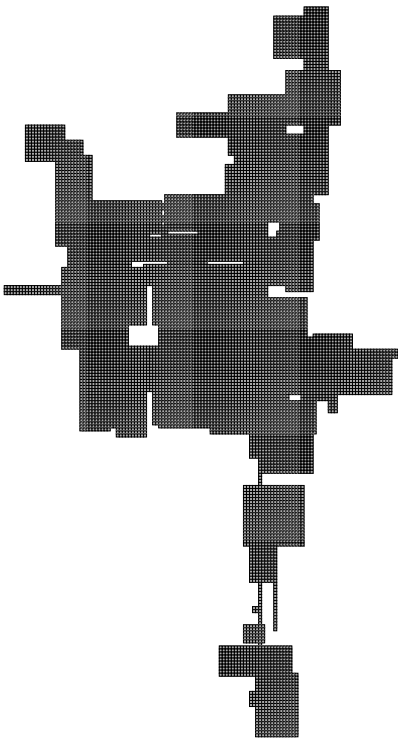

Figure 12: An example instance of Type IIa with 10 052 pixels.

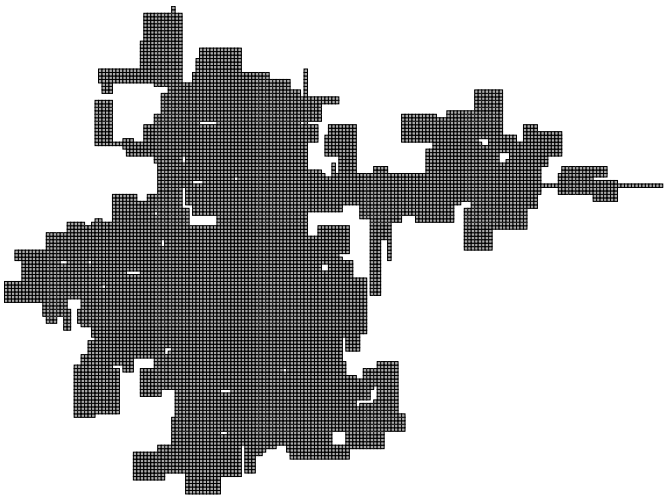

Figure 13: An example instance of Type IIb with 10 004 pixels.

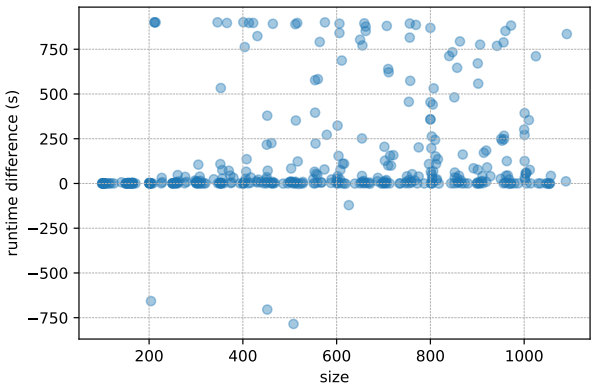

Figure 14: Difference in runtime for cycle cover and tour for the same experiments as in Fig. 15. Runtime capped at 15 min. A positive value means the tour took longer. Surprisingly there are also some instances where the cycle cover was significantly harder to solve (the cycle cover problem is already NP-hard itself). Usually the runtime is very close.

instances within the time limit even though the first is usually faster. A possible explanation may be that the turn cost constraint tends to minimize the number of cycles and usually only few additional constraints are needed. The runtime difference for cycle cover and tours is shown in Fig. 14. It shows that the cycle cover and the tour problem are often nearly equally hard to solve.

For full cycle covers and tours, Fig. 16 gives a comparison and ground truth of our approximation algorithm with the provably optimal objective values for instance sizes up to about 1000 pixels; it can be seen that the computed value is mostly within 1-2% of the optimum. For large instances of Type I and size up to 300 000 pixels, Fig. 18 (left) shows the optimality gap of the computed solutions with respect to the fractional lower bound, both for our new approximation method (FK) and the one by Arkin et al. (ABD+); it can be seen that overall, FK closes about 70% of the gap left by ABD+. Analogous results for Type IIa are shown in Fig. 18 (middle), with easily 75% of the optimality gap of ABD+ closed by FK. The results for Type IIb (plotted in Fig. 18 right) show greater variance, but overall a similar relative performance. A runtime comparison for our approximation algorithm for cycle cover is given in Fig. 17. For some instances of Type II (in particular, for those of Type IIb with the "roughest" boundaries), the runtime is affected by the effort for solving the linear program and computing the matching (as shown in Fig. 17). While a higher turn cost seems to affect the runtime negatively, this effect is low compared to the general deviation. Our optimized implementation of our new approximation method (FK) was frequently faster than our naïve implementation of the approximation algorithm by Arkin et al. (ABD+); however, this is an unfair comparison and an optimized implementation of the later should be visibly faster such that we only have an advantage regarding the solution quality.

Finally, results for the penalty variants of optimal cycle covers and tours for instances of Type IIa up to 50.000 pixels are shown in Fig. 19. Despite the more involved objective function and solution space, the optimality gap compared to a fractional lower bound is typically about 20% and never more than 50%, even for the tour version.

6 Conclusion

We have provided significant practical progress on computing provably optimal and near-optimal solutions to large and very large instances of tour planning problems with turn cost, pushing the frontiers of instances that can be practically handled from the previous 76 pixels to more than 1000 (for provably optimal) and even to more than 300,000 (for near-optimal). As a consequence, practical problems of precision farming and pest control that were previously relying on local heuristics without any performance guarantee can now rely on well-understood algorithmic techniques with excellent performance guarantees. This shows the power of algorithm engineering techniques, demonstrating that new theoretical algorithmic insights can be turned into important practical breakthroughs.

We see a considerable range of practical future developments. These include adapting our techniques to further practical problems, in which covering tours are one component of integrated optimization problems involving several types of agricultural robots. In addition, new progress in precision farming (which aims at making use of refined data describing intricacies of plant growth and ground yield) will make it relevant to adjust subset and penalty tours according to changing situations. We are optimistic that demonstrating the practical benefits of algorithm engineering will be helpful for future interdisciplinary collaboration in the context of digital agriculture.

Acknowledgements

We thank Aaron T. Becker, An Nguyen, Mary Burbage, and Shriya Bhatnagar for the mosquito hunting drone, and Lennart Trösken for helping with the field example. We also thank the anonymous reviewers for their constructive feedback.

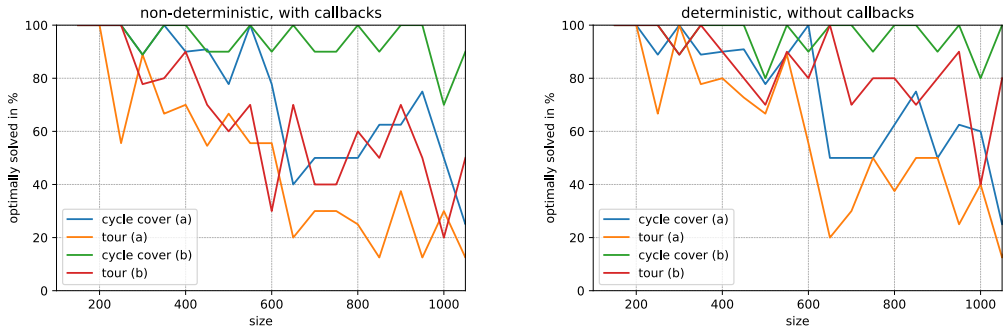

Figure 15: Percentage of instances that could be solved to optimality within a time limit of 15 min. There are roughly 20 instances of Type II(a and b) per 50 pixel step. The cost function is number of transitions plus 50× the number of simple turns. Due to the limited size, the turn costs dominate. The right plot shows a solver that added subtour elimination constraints only to optimal solutions while the left solver used the non-deterministic mode with callbacks that directly added subtour elimination constraints to every integral solution. Suprisingly the later can solve fewer instances to optimality but is usually faster if it can. In both cases the instances with smoother boundary (Type IIa) are harder to solve.

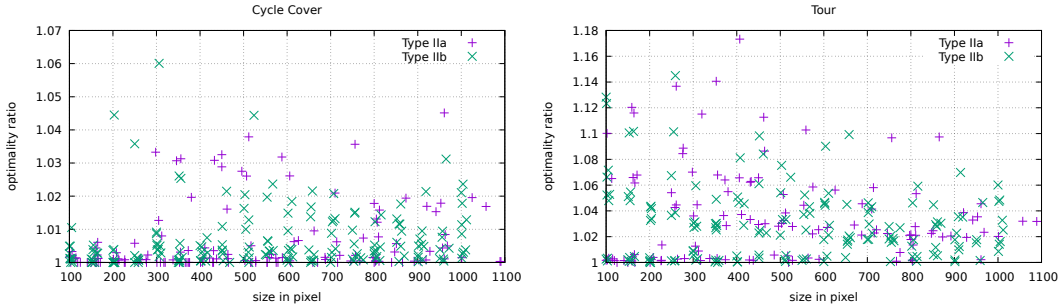

Figure 16: Relative performance of the new approximation algorithm compared to the optimal solution, for instances of Type IIa and Type IIb. For cycle covers (left) it can be seen that the approximation algorithm mostly produces very small gaps for cycle cover; it often finds the optimum, most of the other times it is within less than 1-2% of the optimum. This indicates that for larger instances, a major part of the remaining gap may be due to the heuristic lower bound, implying that the quality of the found solution is even better. For tours (right), the performance is worse, due to heuristic greedy merging of cycles into a tour. The objective function is a linear combination of 50 times the number of simple turns, plus the number of pixel transitions. Only instances that could be solved within 15 min to optimality by the integer program are considered.

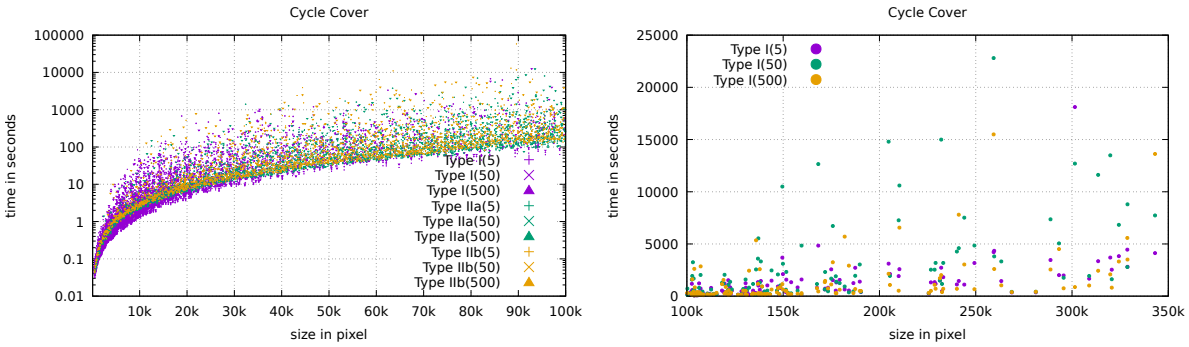

Figure 17: Runtime of the cycle cover approximation algorithm for Type I, Type IIa, Type IIb and different turn cost coefficients, indicating the relative cost of a simple turn vs. a pixel transition. **(Left)** Instances with up to 100 000 pixels for all types. **(Right)** Very large instances with more than 300 000 pixels for Type I.

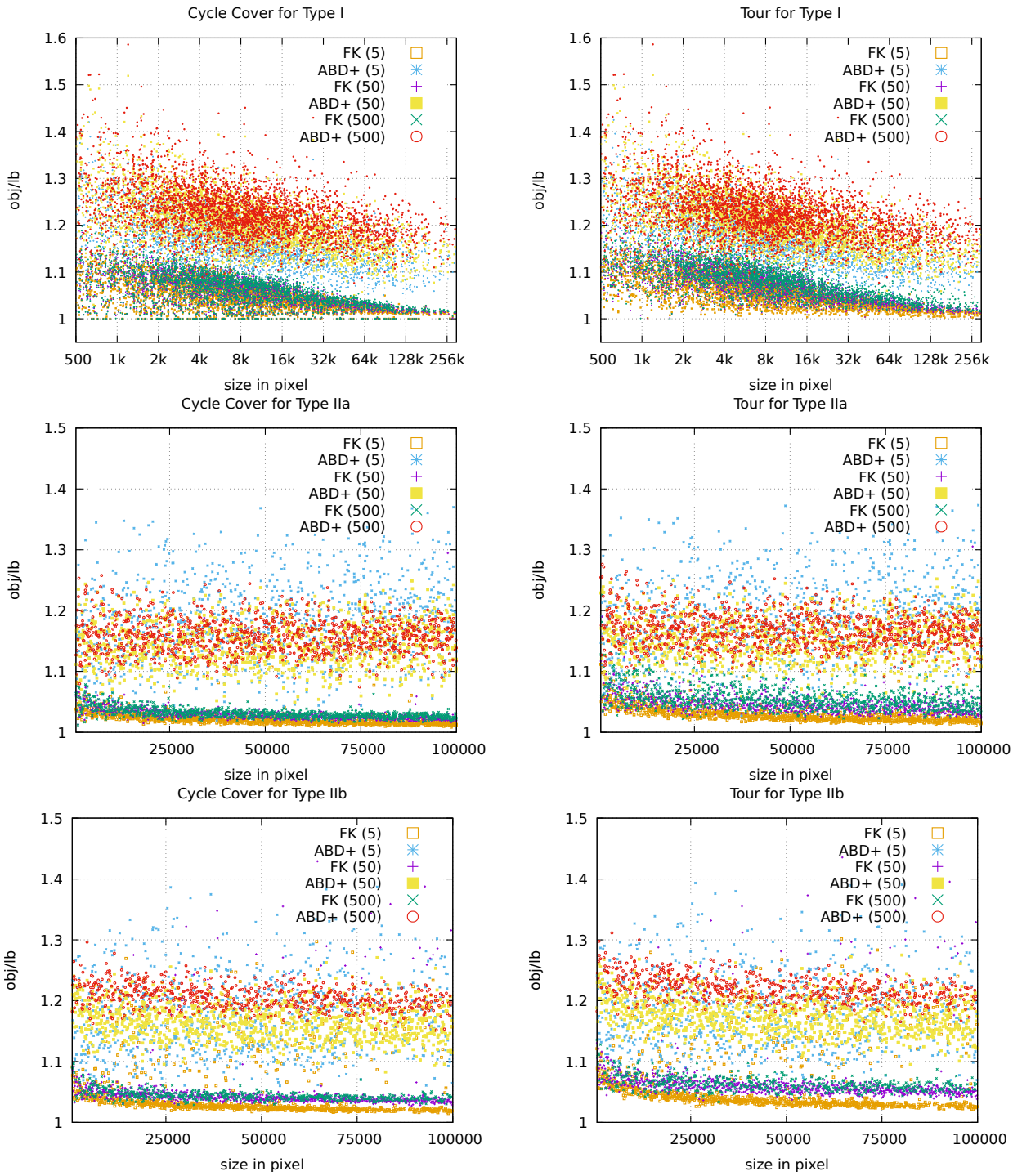

Figure 18: Relative performance of the approximation methods for very large (but simple) instances of Type I and large (and more complex) instances of Type IIa and Type IIb. For examples of these instances see Fig. 11, Fig. 12, resp. Fig. 13. The objective function is a linear combination of c times the number of simple turns, plus the number of pixel transitions, for $c = 5, 50, 500$. Shown is the ratio between the achieved objective value divided by the fractional lower bound. The difference between the cycle cover version and the tour version is in most cases insignificant, because the cycles are usually relatively large, so the cost of connecting the cycles is negligible compared to the costs of the cycles themselves.

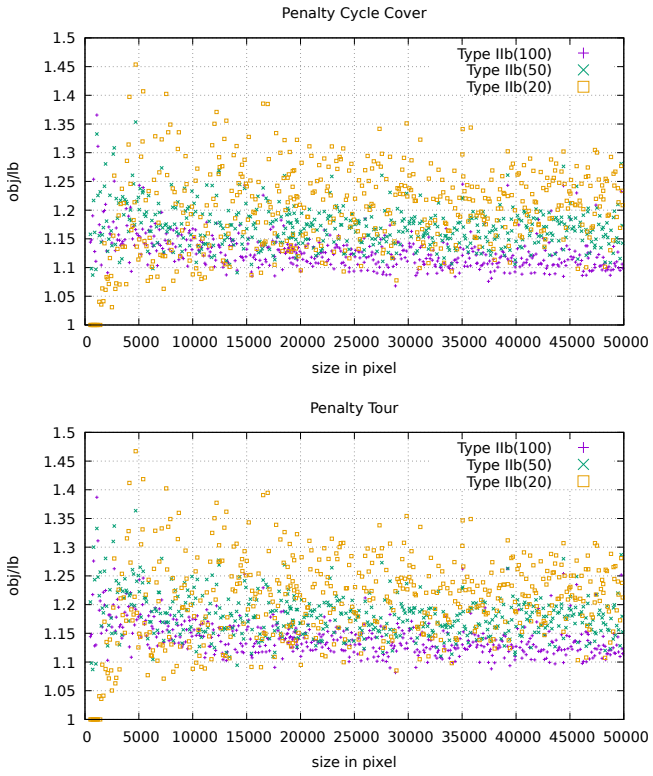

Figure 19: Performance of the penalty variant of the new approximation algorithm on instances of Type IIa. The cost of a simple turn is 500× that of a pixel transition, while the penalty for missing a pixel is 100, 50 or 20 times the cost of a pixel transition. The 'pockets' at the boundaries make instances of this type particularly challenging, because paying the penalty instead of performing expensive detours is a non-trivial alternative.

References

[1] Amit Agarwal, Meng-Hiot Lim, Meng-Joo Er, and Chan Yee Chew. Aco for a new TSP in region coverage. In *Proceedings of the 18th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1717–1722, 2005.

[2] Pankaj K. Agarwal, Therese C. Biedl, Sylvain Lazard, Steve Robbins, Subhash Suri, and Sue Whitesides. Curvature-constrained shortest paths in a convex polygon. *SIAM Journal on Computing*, 31(6):1814–1851, 2002.

[3] Pankaj K. Agarwal and Hongyan Wang. Approximation algorithms for curvature-constrained shortest paths. *SIAM Journal on Computing*, 30(6):1739–1772, 2000.

[4] Alok Aggarwal, Don Coppersmith, Sanjeev Khanna, Rajeev Motwani, and Baruch Schieber. The angular-metric traveling salesman problem. *SIAM Journal on Computing*, 29(3):697–711, 1999.

[5] Ali Ahmadzadeh, James Keller, George Pappas, Ali Jadbabaie, and Vijay Kumar. An optimization-based approach to time-critical cooperative surveillance and coverage with uavs. In *Experimental Robotics*, pages 491–500. Springer, 2008.

[6] Oswin Aichholzer, Anja Fischer, Frank Fischer, J Fabian Meier, Ulrich Pferschy, Alexander Pilz, and Rostislav Staněk. Minimization and maximization versions of the quadratic travelling salesman problem. *Optimization*, 66(4):521–546, 2017.

[7] Osman Ali, Bart Verlinden, and Dirk Van Oudheusden. Infield logistics planning for crop-harvesting operations. *Engineering Optimization*, 41(2):183–197, 2009.

[8] David L. Applegate, Robert E. Bixby, Vašek Chvátal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2007.

[9] Esther M. Arkin, Michael A. Bender, Erik D. Demaine, Sándor P. Fekete, Joseph S. B. Mitchell, and Saurabh Sethia. Optimal covering tours with turn costs. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 138–147, 2001.

[10] Esther M. Arkin, Michael A. Bender, Erik D. Demaine, Sándor P. Fekete, Joseph S. B. Mitchell, and Saurabh Sethia. Optimal covering tours with turn costs. *SIAM Journal on Computing*, 35(3):531–566, 2005.

[11] Esther M. Arkin, Sándor P. Fekete, and Joseph S. B. Mitchell. Approximation algorithms for lawn mowing and milling. *Computational Geometry: Theory and Applications*, 17(1-2):25–50, 2000.

[12] Aaron T. Becker, Moustafa Debboun, Sándor P. Fekete, Dominik Krupke, and An Nguyen. Zapping zika with a mosquito-managing drone: Computing optimal flight patterns with minimum turn cost. In *Proceedings of the 33rd International Symposium on Computational Geometry (SoCG)*, pages 62:1–62:5, 2017. Video at `https://www.youtube.com/watch?v=SFyOMDgdNao`.

[13] Jean-Daniel Boissonnat and Sylvain Lazard. A polynomial-time algorithm for computing a shortest

path of bounded curvature amidst moderate obstacles. In *Proceedings of the 12th Annual Symposium on Computational Geometry (SoCG)*, pages 242–251, 1996.

[14] G.B. Dantzig, D.R. Fulkerson, and S.M. Johnson. Solution of a large-scale traveling-salesman problem. *Operations Research*, 2:393—410, 1954.

[15] Igor R. de Assis and Cid C. de Souza. Experimental evaluation of algorithms for the orthogonal milling problem with turn costs. In *Proceedings of the 10th International Symposium on Experimental Algorithms (SEA)*, pages 304–314, 2011.

[16] Erik D. Demaine, Joseph S. B. Mitchell, and O'Rourke Joseph. The open problems project. URL: `http://cs.smith.edu/~orourke/TOPP/`.

[17] Lester E Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516, 1957.

[18] Sándor P. Fekete and Dominik Krupke. Covering tours and cycle covers with turn costs: Hardness and approximation. *arXiv*, 2018. http://arxiv.org/abs/1808.04417.

[19] Sándor P. Fekete and Gerhard J. Woeginger. Angle-restricted tours in the plane. *Computational Geometry: Theory and Applications.*, 8:195–218, 1997.

[20] Yoav Gabriely and Elon Rimon. Spiral-stc: An on-line coverage algorithm of grid environments by a mobile robot. In *Proceedings of the 19th IEEE Conference on Robotics and Automation (ICRA)*, pages 954–960. IEEE, 2002.

[21] Enric Galceran and Marc Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12):1258–1276, 2013.

[22] Chinmay Hegde, Piotr Indyk, and Ludwig Schmidt. A nearly-linear time framework for graph-structured sparsity. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4165–4169, 2016.

[23] Wesley H. Huang. Optimal line-sweep-based decompositions for coverage algorithms. In *Proceedings of the 18th IEEE Conference on Robotics and Automation (ICRA)*, pages 27–32. IEEE, 2001.

[24] Gerold Jäger and Paul Molitor. Algorithms and experimental study for the traveling salesman problem of second order. In *Proceedings of the 2nd International Conference on Combinatorial Optimization and Applications (COCOA)*, pages 211–224. 2008.

[25] Vladimir Kolmogorov. Blossom V: A new implementation of a minimum cost perfect matching algorithm. *Mathematical Programming Computation*, 1(1):43–67, Jul 2009.

[26] Sylvain Lazard, John Reif, and Hongyan Wang. The complexity of the two dimensional curvature-constrained shortest-path problem. In *Proceedings of the 3rd International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, pages 49–57, 1998.

[27] Olaf Maurer. Winkelminimierung bei Überdeckungsproblemen in Graphen. Diplomarbeit, Technische Universität Berlin, 2009.

[28] An Nguyen, Dominik Krupke, Mary Burbage, Shriya Bhatnagar, Sándor P. Fekete, and Aaron T. Becker. Using a UAV for destructive surveys of mosquito population. In *Proceedings of the 35th IEEE Conference on Robotics and Automation (ICRA)*, pages 7812–7819, 2018.

[29] Simeon C. Ntafos. Watchman routes under limited visibility. *Computational Geometry: Theory and Applications.*, 1:149–170, 1991.

[30] Jerome Le Ny, Eric Feron, and Emilio Frazzoli. On the Dubins traveling salesman problem. *IEEE Transactions on Automation and Control*, 57(1):265–270, 2012.

[31] Borzou Rostami, Francesco Malucelli, Pietro Belotti, and Stefano Gualandi. Quadratic TSP: A lower bounding procedure and a column generation approach. In *Proceedings of the IEEE Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 377–384, 2013.

[32] Ryo Takei, Richard Tsai, Haochong Shen, and Yanina Landa. A practical path-planning algorithm for a simple car: A Hamilton-Jacobi approach. In *Proceedings of the 29th American Control Conference (ACC)*, pages 6175–6180, 2010.

[33] Zhiyang Yao, Satyandra K Gupta, and Dana S Nau. Hybrid cutter path generation for 2.5D milling operation. In *Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE)*, pages 703–714, 2002.

[34] Alexander Zelinsky, Ray A Jarvis, JC Byrne, and Shin'ichi Yuta. Planning paths of complete coverage of an unstructured environment by a mobile robot. In *Proceedings of the 6th International Conference on Advanced Robotics (ICAR)*, pages 533–538, 1993.