

Covering Tours and Cycle Covers with Turn Costs: Hardness and Approximation

Sándor P. Fekete¹ and Dominik Krupke¹

¹ Department of Computer Science, TU Braunschweig, 38106 Braunschweig, Germany
 {s.fekete,d.krupke}@tu-bs.de

Abstract

We investigate a variety of problems of finding tours and cycle covers with minimum turn cost. Questions of this type have been studied in the past, with complexity and approximation results, and open problems dating back to work by Arkin et al. in 2001. A wide spectrum of practical applications have renewed the interest in these questions, and spawned variants: for *full coverage*, every point has to be covered, for *subset coverage*, specific points have to be covered, and for *penalty coverage*, points may be left uncovered by incurring an individual penalty.

We make a number of contributions. We first show that finding a minimum-turn (full) cycle cover is NP-hard even in 2-dimensional grid graphs, solving the long-standing open *Problem 53* in *The Open Problems Project* edited by Demaine, Mitchell and O'Rourke. We also prove NP-hardness of finding a *subset* cycle cover of minimum turn cost in *thin* grid graphs, for which Arkin et al. gave a polynomial-time algorithm for full coverage; this shows that their boundary techniques cannot be applied to compute exact solutions for subset and penalty variants.

On the positive side, we establish the first constant-factor approximation algorithms for all considered subset and penalty problem variants for very general classes of instances, making use of LP/IP techniques. For these generalized graph problems with many possible edge directions (and thus, turn angles, such as in hexagonal grids or higher-dimensional variants), our approximation factors also improve the combinatorial ones of Arkin et al. Our approach can also be extended to other geometric variants, such as scenarios with obstacles and linear combinations of turn and distance costs.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

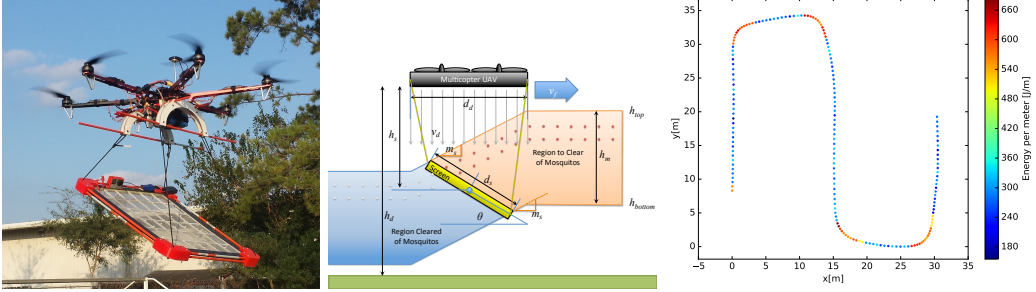
Keywords and phrases Geometric optimization, optimal tours, optimal cycle covers, turn cost, NP-hardness, approximation

1 Introduction

Finding roundtrips of minimum cost is one of the classic problems of theoretical computer science. In its most basic form, the objective of the *Traveling Salesman Problem (TSP)* is to minimize the total length of a single tour that covers all of a given set of locations. If the tour is not required to be connected, the result may be a *cycle cover*: a set of closed subtours that together cover the whole set. This distinction makes a tremendous difference for the computational complexity: while the TSP is NP-hard, computing a cycle cover of minimum total length can be achieved in polynomial time, based on matching techniques.

Evaluating the cost for a tour or a cycle cover by only considering its length may not always be the right measure. Fig. 1 shows an example application, in which a drone has to sweep a given region to fight mosquitoes that may transmit dangerous diseases. As can be

seen in the right-hand part of the figure, by far the dominant part of the overall travel cost occurs when the drone has to change its direction. (See our related video and abstract [11] for more details, and the resulting tour optimization.) There is an abundance of other related applied work, e.g., mowing lawns or moving huge wind turbines [9].



■ **Figure 1 (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 [11] for an accompanying abstract.

For many purposes, two other variants are also practically important: for *subset coverage*, only a prespecified subset of locations needs to be visited, while for *penalty coverage*, locations may be skipped at the expense of an individual penalty. From the theoretical side, Arkin et al. [6] showed that finding minimum-turn tours in grid graphs is NP-hard, even if a minimum-turn cycle cover is given. The question whether a minimum-turn cycle cover can be computed in polynomial time (just like a minimum-length cycle cover) has been open for at least 17 years, dating back to the conference paper [5]; it has been listed for 15 years as *Problem 53* in *The Open Problems Project* edited by Demaine, Mitchell, and O’Rourke [17]. In Section 2 we resolve this problem by showing that computing a minimum-turn cycle cover in planar grid graphs is indeed NP-hard.

This makes it important to develop approximation algorithms. In Section 3, we present a general technique based on Integer Programming (IP) formulations and their Linear Programming (LP) relaxations. By the use of polyhedral results and combinatorial modifications, we can establish constant approximation factors of this technique for all problem variants.

1.1 Related Work

Milling with Turn Costs. Arkin et al. [5, 6] introduce the problem of milling (i.e., “carving out”) with turn costs. They show hardness of finding an optimal tour, even in *thin* 2-dimensional grid graphs (which do not contain an induced 2×2 subgraph) with a given optimal cycle cover. They give a 2.5-approximation algorithm for obtaining a cycle cover, resulting in a 3.75-approximation algorithm for tours. The complexity of finding an optimal cycle cover in a 2-dimensional grid graph was established as *Problem 53* in *The Open Problems Project* [17].

Maurer [24] proves that a cycle *partition* with a minimum number of turns in grid graphs can be computed in polynomial time and performs practical experiments for optimal cycle covers. De Assis and de Souza [15] computed a provably optimal solution for an instance with 76 vertices. For the abstract version on graphs, Fellows et al. [21] show that the problem is fixed-parameter tractable by the number of turns, tree-width, and maximum degree. Benbernou [12] considered milling with turn costs on the surface of polyhedrons in the 3-dimensional grid. She gives a corresponding $8/3$ -approximation algorithm for tours.

Note that the theoretical work presented in this paper has significant practical implications. As described in our forthcoming conference paper [19], the IP/LP-characterization presented in Section 3 can be modified and combined with additional algorithm engineering techniques to allow solving instances with more than 1000 pixels to provable optimality (thereby expanding the range of de Assis and de Souza [15] by a factor of 15), and computing solutions for instances with up to 300,000 pixels within a few percentage points (thereby showing that the practical performance of our approximation techniques is dramatically better than the established worst-case bounds).

For mowing problems, i.e., covering a given area with a moving object that may leave the region, Stein and Wagner [27] give a 2-approximation algorithm on the number of turns for the case of orthogonal movement. If only the traveled distance is considered, Arkin et al. [7] provide approximation algorithms for milling and mowing.

Angle and curvature-constrained tours and paths. If the instances are in the \mathbb{R}^2 plane and only the turning angles are measured, the problem is called the *Angular Metric Traveling Salesman Problem*. Aggarwal et al. [3] prove hardness and provide an $O(\log n)$ approximation algorithm for cycle covers and tours that works even for distance costs and higher dimensions. As shown by Aichholzer et al. [4], this problem seems to be very hard to solve optimally with integer programming. Fekete and Woeginger [20] 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. [23]. Without obstacles, the problem is known as the *Dubins path* [18] that can be computed efficiently. For different types of obstacles, Boissonnat and Lazard [13], Agarwal et al. [1] and Agarwal and Wang [2] provide polynomial-time algorithms or $1 + \epsilon$ approximation algorithms, respectively. Takei et al. [28] consider the solution of the problem from a practical perspective. The *Dubins Traveling Salesman Problem* is considered by Le Ny et al. [25].

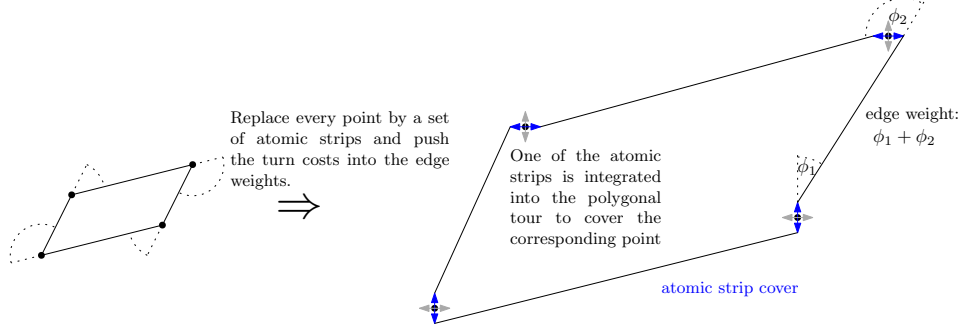
Related combinatorial problems. Goemans and Williamson [22] provide an approximation technique for constrained forest problems and similar problems that deal with penalties. In particular, they provide a 2-approximation algorithm for *Prize-Collecting Steiner Trees* in general symmetric graphs and the *Penalty Traveling Salesman Problem* in graphs that satisfy the triangle inequality. An introduction into approximation algorithms for prize-collecting/penalty problems, k-MST/TSP, and minimum latency problems is given by Ausiello et al. [10].

1.2 Preliminaries

The angular metric traveling salesman problem resp. cycle cover problem ask for a cycle resp. set of cycles such that a given set P of n points in \mathbb{R}^d is covered and the sum of turn angles is minimized. A cycle is a closed chain of segments and covers the points of the segments' joints (at least two of P). The turn angle of a joint is the angle difference to 180° . In the presence of polygonal obstacles, cycles are not allowed to cross them. We consider three coverage variants: Full, subset, and penalty. In full coverage, every point has to be covered. In subset coverage, only points in a subset $S \subseteq P$ have to be covered (which is only interesting for grid graphs). In penalty coverage, no point has to be covered but every uncovered point $p \in P$ induces a penalty $c(p) \in \mathbb{Q}_0^+$ on the objective value. Optionally, the objective function can be a linear combination of distance and turn costs.

In the following, we introduce the *discretized angular metric*, by considering for every point $p \in P$ a set of ω possible orientations (and thus, 2ω possible directions) for a trajectory through p . We model this by considering for each $p \in P$ a set O_p of ω infinitely short

segments, which we call *atomic strips*; a point is covered if one of its segments is part of the cycle, see Fig. 2. The corresponding selection of atomic strips is called *Atomic Strip Cover*, i.e., a selection of one $o \in O_p$ for every $p \in P$.



■ **Figure 2** Transforming an angular metric TSP instance and solution to an instance based on atomic strips, which can be considered infinitely small segments.

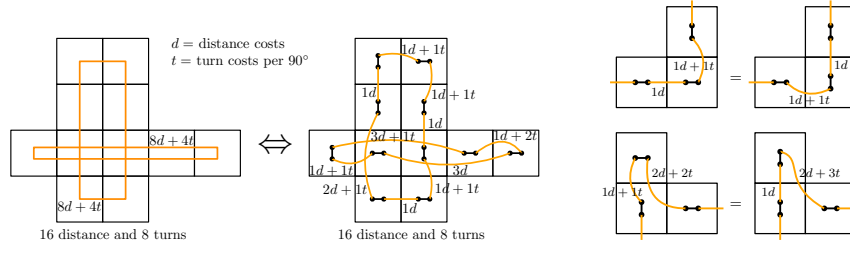
The atomic strips induce a weighted graph $G_O(V_O, E_O)$ with the endpoints of the atomic strips as vertices and the connections between the endpoints as edges. The weight of an edge in G_O equals the connection costs, in particular the turn costs on the two endpoints. Thus, the cycle cover problem turns into finding an Atomic Strip Cover with the minimum-weight perfect matching on its induced subgraph. As the cost of connections in it depends on *two* edges in the original graph, we call this generalized problem (in which the edge weights do not have to be induced by geometry) the *semi-quadratic cycle cover problem*.

It is important to note that the weights do not satisfy the triangle inequality; however, a direct connection is not more expensive than a connection that includes another atomic strip, giving rise to the following *pseudo-triangle inequalities*.

$$\forall v_1, v_2 \in V_O, w_1 w_2 \in O_p, p \in P : \begin{aligned} \text{cost}(v_1 v_2) &\leq \text{cost}(v_1 w_1) + \text{cost}(w_2 v_2) \\ \text{cost}(v_1 v_2) &\leq \text{cost}(v_1 w_2) + \text{cost}(w_1 v_2) \end{aligned} \quad (1)$$

Our model allows the original objective function to be a linear combination of turn and distance costs, as it does not influence Eq. (1). Instances with polygonal obstacles for 2-dimensional geometric instances are also possible (however, for 3D, the corresponding edge weights can no longer be computed efficiently). A notable special case are *grid graphs* that arise as vertex-induced subgraph of the infinite integer orthogonal grid. In this case, a point can only be covered straight, by a simple 90° turn, or by a 180° u-turn. We show grid graphs as polyominoes in which vertices are shown as *pixels*. We also speak of the number of *simple turns* (u-turns counting as two) instead of turn angles. More general grid graphs can be based on other grids, such as 3-dimensional integral or hexagonal grids.

Minimum turn cycle covers in grid graphs can be modelled as a semi-quadratic cycle cover problem with $\omega = 2$ and edge weights satisfying Eq. (1). One of the atomic strips represents being in a horizontal orientation (with an east and a west heading vertex) and the other being in a vertical orientation (with a north and a south heading vertex). The cost of an edge is as follows: Every vertex is connected to a position and a direction. The cost is the cheapest transition from the position and direction of the first vertex to the position and opposite heading of the second vertex (this is symmetric and can be computed efficiently). We can easily transform a cycle cover in a grid graph into one based on atomic strips and vice versa, see Fig. 3 (left). For each pixel we choose one of its transitions. If it is straight, we select the equally oriented strip; otherwise it does not matter, see Fig. 3 (right). With more



■ **Figure 3 (Left)** From an optimal cycle cover (dotted) we can extract an Atomic Strip Cover (thick black), such that the matching (orange) induces an optimal solution. **(Right)** For turns it does not matter if we choose the horizontal or vertical atomic strip.

atomic strips we can also model more general grid graphs such as hexagonal or 3-dimensional grid graphs with three atomic strips.

1.3 Our Contribution

We provide the following results.

- We resolve *Problem 53* in *The Open Problems Project* [17] by proving that finding a cycle cover of minimum turn cost is NP-hard, even in the restricted case of grid graphs. We also prove 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 [6].
- We provide a general IP/LP-based technique for obtaining $2 * \omega$ approximations for the semi-quadratic (penalty) cycle cover problem if Eq. (1) is satisfied, where ω is the maximum number of atomic strips per vertex.
- We show how to connect the cycle covers to minimum turn tours to obtain a 6 approximation for full coverage in regular grid graphs, 4ω approximations for full tours in general grid graphs, $4\omega + 2$ approximations for (subset) tours, and $4\omega + 4$ for penalty tours.

To the best of our knowledge, this is the first approximation algorithm for the subset and penalty variant with turn costs. For general grid graphs our techniques yields better guarantees than the techniques of Arkin et al. who give a factor of $6 * \omega$ for cycle covers and $6 * \omega + 2$ for tours. In practice, our approach also yields better solutions for regular grid graphs, see [19].





2 Complexity

Problem 53 in *The Open Problems Project* asks for the complexity of finding a minimum-turn (full) cycle cover in a 2-dimensional grid graph. This is by no means obvious: large parts of a solution can usually be deduced by local information and matching techniques. In fact, it was shown by Arkin et al. [5, 6] that the full coverage variant in *thin* grid graphs (which do not contain a 2×2 square, so every pixel is a boundary pixel) is solvable in polynomial time. In this section, we prove that finding a *full* cycle cover in 2-dimensional grid graphs with minimum turn cost is NP-hard, resolving *Problem 53*. We also show that *subset* coverage is NP-hard even for *thin* grid graphs, so the boundary techniques by Arkin et al. [5, 6] do not provide a polynomial-time algorithm.

2.1 Full Coverage in Grid Graphs

► **Theorem 1.** *It is NP-hard to find a cycle cover with a minimum number of 90° turns (180° turns counting as two) in a grid graph.*

The proof is based on a reduction from *One-in-three 3SAT* (1-in-3SAT), which was shown to be NP-hard by Schaefer [26]: for a Boolean formula in conjunctive normal form with only three literals per clause, decide whether there is a truth assignment that makes exactly one literal per clause **true** (and exactly two literals **false**). For example, $(x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$ is not (1-in-3) satisfiable, whereas $(x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_4})$ is satisfiable.

An example is given in Fig. 4 for the instance $x_1 + x_2 + x_3 = 1 \wedge \overline{x_1} + \overline{x_2} + \overline{x_4} = 1 \wedge \overline{x_1} + x_2 + \overline{x_3} = 1$. For every variable we have a  gadget consisting of a gray  gadget and a zig-zagging, high-cost path of blue pixels. A cheap solution traverses a blue path once and connect the ends through the remaining construction of gray and red pixels. Such *variable cycles* (highlighted in red) must either go through the upper () or lower () lane of the variable gadget; the former corresponds to a **true**, the later to a **false** assignment of the corresponding variable. A *clause gadget* modifies a lane of all three involved variable gadgets. This involves the gray pixels that are covered by the green cycles; we can show that they do not interfere with the cycles for covering the blue and red pixels, and cannot be modified to cover them. Thus, we only have to cover red and blue pixels, but can pass over gray pixels, too.

To this end, we must connect the ends of the blue paths; as it turns out, the formula is satisfiable if and only if we can perform this connection in a manner that also covers one corresponding red pixel with at most two extra turns.

There are three important conclusions.

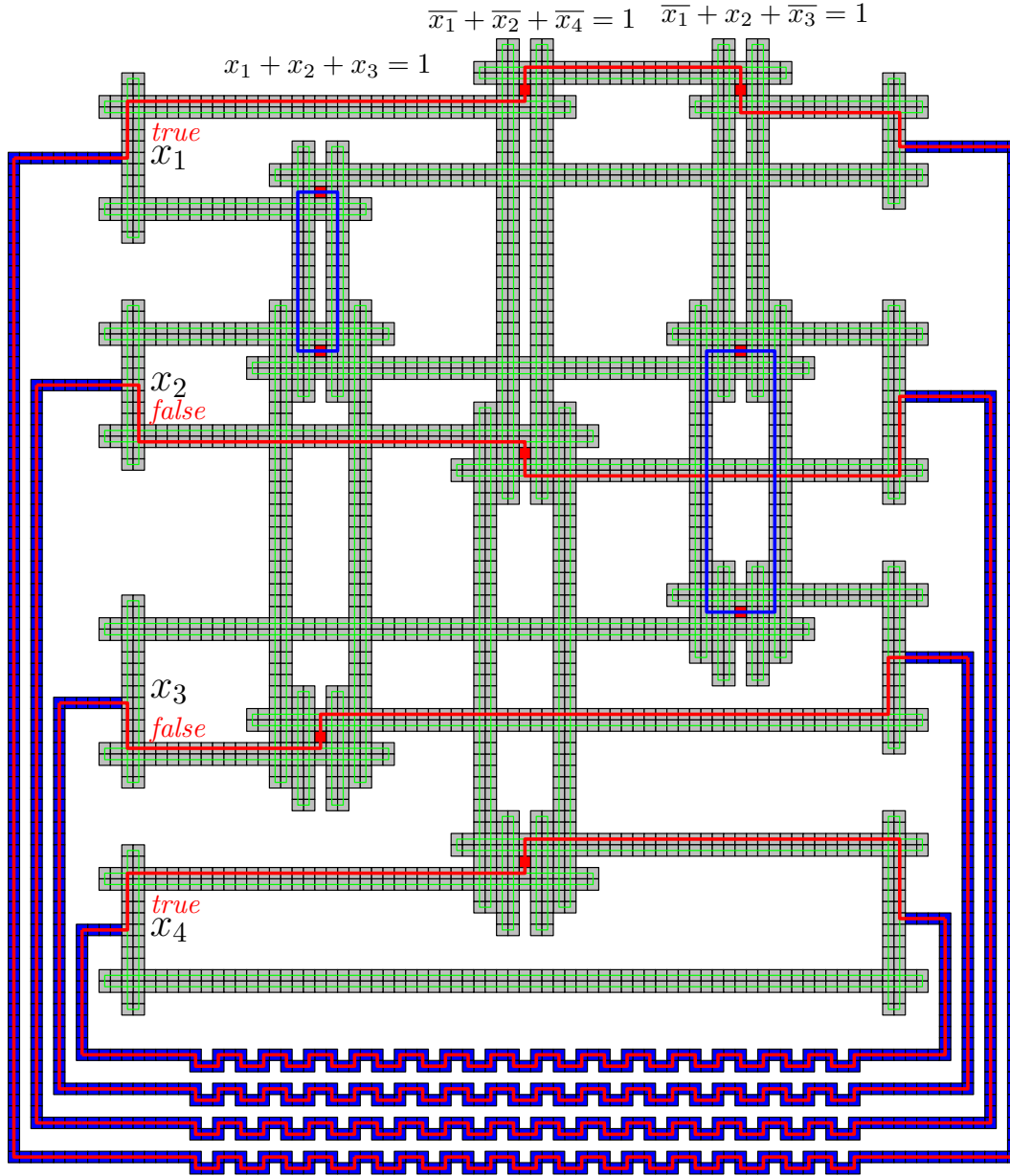
1. The gray pixels are always covered by the green cycles or the solution can be modified without extra turns. Thus, we only need to focus on the logic of covering the blue and red pixels. This is based on the observation that for keeping tight local upper bounds on the number of turns, we are only allowed to make turns at very specific locations.
2. There are only the red variable cycles that cover exactly the red pixels on the **true** or **false** lane and the simple 4-turn blue cycles for covering two vertically adjacent red pixels available for covering the red and blue pixels within the upper bounds of turns for satisfiable constructions.
3. A coverage with only two turns per red pixel is possible if and only if the underlying formula is satisfiable.

The first item is the most challenging, but also most crucial part.

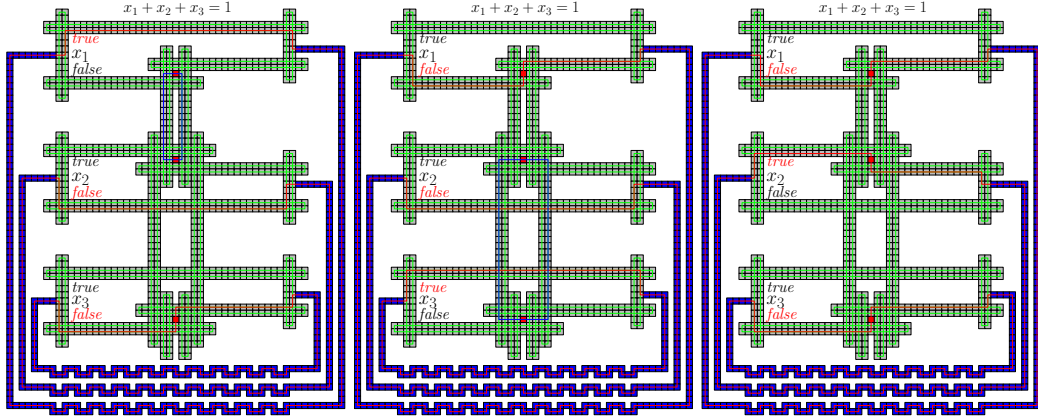
2.1.1 Basic Construction

We use a reduction from *One-in-three 3SAT*. Refer to Fig. 4 for the overall construction. Fig. 6 shows the building block for a variable gadget, formed by a zig-zagging path, marked in blue; this path enforces its covering cycle to go through the box (gray) of the variable, because a double coverage of itself would be too expensive because of the zig-zagging part. It can be seen that this covering cycle takes the upper path if the corresponding variable is set to **true** and takes the lower path if the variable is set to **false**.

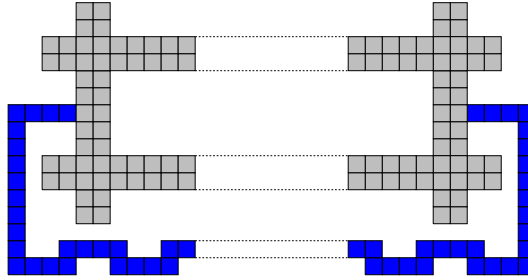
The basic idea for representing literals in clauses is shown in Fig. 7; this basic element is employed in sets of three for each clause, arranged in a horizontal vector of “connectors” to the corresponding variable paths, making use of appropriate geometric extensions to account for (logical) parity, which are described and motivated in the following subsection; of critical importance is the central “crucial” pixel marked in red in all figures.



■ **Figure 4** Representing the 1-in-3SAT-formula $x_1 + x_2 + x_3 = 1 \wedge \overline{x_1} + \overline{x_2} + \overline{x_4} = 1 \wedge \overline{x_1} + x_2 + \overline{x_3} = 1$.



■ **Figure 5** Construction for the one-clause formula $x_1 + x_2 + x_3 = 1$ and three possible solutions. Every variable has a cycle traversing a zig-zagging path of blue pixels. A variable is **true** if its cycle uses the upper path through green/red pixels, **false** if it takes the lower path. For covering the red pixels, we may use two additional turns. This results in three classes of optimal cycle covers, shown above. If we use the blue 4-turn cycle to cover the upper two red pixels, we are forced to cover the lower red pixel by the x_3 variable cycle, setting x_3 to **false**. The variable cycles of x_1 and x_2 take the cheapest paths, setting them to **true** or **false**, respectively. The alternative to a blue cycle is to cover all three red pixel by the variable cycles, as in the right solution.

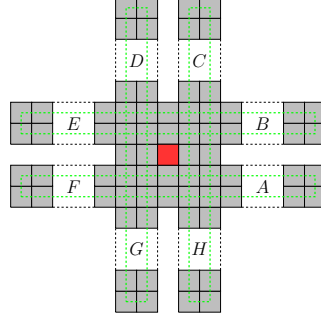


■ **Figure 6** The construction of a variable gadget. The blue part has to be covered by letting the cycle corresponding to this variable go through the gray part. If the path takes the upper path, the variable is set to **true**; if it takes the lower path, the variable is set to **false**.

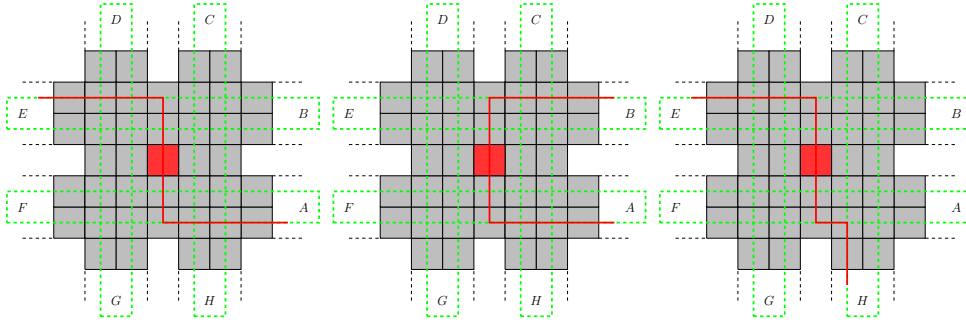
2.1.2 Logical Ideas and Details of the Construction

The key idea of the reduction is that information can only be encoded in pixels for which we do not know in which orientation they are crossed without turning (otherwise matching techniques can be used, as utilized by our approximation algorithm). This means that, e.g., pixels on the boundary are easy to deal with. Also, pixels for which we know that there is a turn or that are crossed straight in both orientations are not critical: in a solution, they may only be used indirectly for information propagation. Based on these observations, we consider variants of the construction in Fig. 7 as fundamental information elements. The optimality of the green cycles is argued in the next section, and thus the corresponding pixels are trivially covered.

Covering a crucial pixel is more difficult. Covering it by an additional cycle or modifying the green cycles costs always at least 4 turns. Assume we create a path from one entry (A, B, C, D, E, F, G, H) to another and cover the crucial pixel by this path. For example, a path entering at A and leaving at E would allow us to cover the crucial pixel at a cost of 2



■ **Figure 7** Basic block for the hardness proof. Only the red “crucial” pixel is of algorithmic interest, as described in the text.



■ **Figure 8** Three ways of covering the crucial pixel. The first two keep the orientation and have a cost of 2. The third one changes the orientation, and thus has a cost of at least 3.

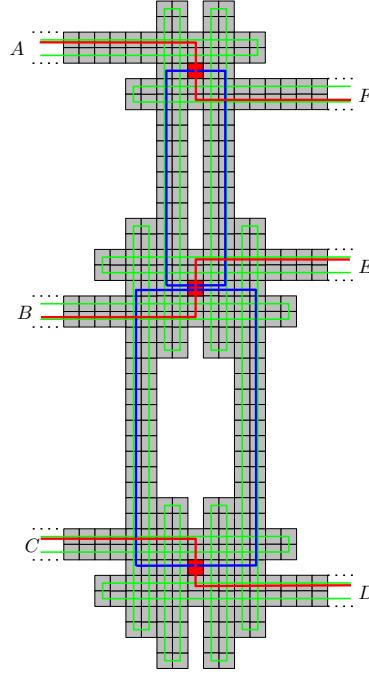
while a path entering at A and leaving at D has at least 3 turns if also covering the crucial pixel, see Fig. 8. Actually, we can cover the crucial pixel with a cost of 2 exactly with S- and U-turns, i.e., paths that do not change the orientation. In addition, it can be seen that the crucial pixel always needs at least two turns within this construction.

Now consider the construction shown in Fig. 9 that uses a combination of elements as described above. Assume all turns made outside the construction are free and all six “exits” A, B, C, D, E, F are connected from outside. This means that every red highlighted pixel can be covered by a cycle of cost two that traverses the exits above and below (see blue paths in Fig. 9). The green cycles are still necessary; employing them for covering the crucial pixel is too expensive, so we can concentrate on how to cover the red highlighted pixels by additional cycles. It can be seen that there are only five potential cycles that involve not more than a cost of two turns per crucial pixel: these are the three mentioned red cycles that use the exterior and the two interior cycles marked in blue in Fig. 9 (the cycles can be slightly shifted without changing them in a useful manner). This results in only three potential optimal solutions (refer to Fig. 5):

1. Using the upper red cycle to cover the upper crucial pixel and the lower blue cycle to cover the middle crucial pixel and lower crucial pixel.
2. Using the lower red cycle to cover the lower crucial pixel and the upper blue cycle to cover the middle crucial pixel and the upper crucial pixel.
3. Using all three red cycles to cover each crucial pixel separately.

For constructing a clause $x_1 + x_2 + x_3 = 1$ with x_1 being above x_2 and x_3 being below x_2 we now place the upper crucial pixel of Fig. 9 on the **false** path of x_1 , the middle pixel

on the **true** path of x_2 and the lower crucial pixel on the **false** path of x_3 as illustrated in Fig. 5. If we take the **true** path of x_2 , we now have to take the **false** paths of x_1 and x_3 . If we take the **false** path of x_2 we need to block the **false** path of either x_1 or x_3 (but not both) by a red cycle. The other variable paths (without the crucial pixel) do not need any turns and are hence always preferred if the red pixel does not have to be covered. Note that any external cycle that enters the construction (e.g., through A) needs at least two turns to leave it again through another exit. Hence, passing through this construction should always be combined with covering an uncovered red pixel, for which we have only one choice with two turns, in order to account for the two necessary turns. By concatenating these clauses, we can form any *One-in-three 3SAT* formula, see Fig. 4.



■ **Figure 9** Assuming that all turns made outside are free, there are only five potential cycles to cover the crucial pixels: the three exterior red cycles and the two interior blue cycles. Assigning the usage of the upper red cycle the Boolean variable x_1 (and analogous for the middle and the lower x_2 and x_3 , resp.) leads to the Boolean formula $(x_1 \vee x_3) \wedge (x_2 \leftrightarrow x_1 \wedge x_3)$ or $\overline{x_1} + x_2 + \overline{x_3} = 1$.

2.1.3 A Tight Lower Bound

We now give a tight lower bound on the number of turns in a cycle cover. In order to achieve it, the turns have to be made at very specific positions. For this purpose it is sufficient to limit our view onto local components and state that in each of these disjunct components a specific number of turns has to be made. The global lower bound is then the sum of all local lower bounds. Later we show that these turns are only sufficient if the corresponding formula is satisfiable.

► **Lemma 2.** *On every full strip (i.e., a maximal connected collinear set of pixels) in the grid graph, a cycle cover has to have an even number of turns (u-turns counting twice).*

Proof. For every turn that enters the strip, there needs to be a turn for leaving it. Turns that do not enter/leave the strip are u-turns. ◀

► **Lemma 3.** *Given any pixel p of a grid graph and a corresponding cycle cover \mathcal{C} . If there is no turn on p , then there has to be a turn left of p and another turn right of p (on the horizontal full strip through p), or there has to be a turn above p and another below p (on the vertical full strip through p). If there is a turn at p , there are at least three turns on the star consisting of the horizontal and vertical full strip through p .*

Proof. If there is no turn on p , there is a straight part of a cycle going through it. At either end of this straight part, it needs to turn in order to close the cycle. If there is a turn at p , the claim follows by Lemma 2. ◀

Now we can deduce the turn positions for all essential parts of the construction, as stated in the next lemma.

► **Lemma 4.** *In Fig. 14, the essential parts of the construction are displayed (possibly reflected). For these parts we can give the following lower bounds on the necessary number of interior turns and constraints on the positions of turns if this bound is tight: In a part as shown in Fig. 10/11/12/13 (excluding the blue area), we need at least 10/18/14/10 interior turns. To achieve exactly this lower bound, there has to be exactly one turn at every black dot and the remaining two turns have to be in the green area.*

Proof. There needs to be at least one turn in every corner pixel, making Fig. 10 and 13 trivial. In Fig. 11 and 12 we have some additional non-corner pixels for which we need easy additional arguments based on Lemma 2 and 3.

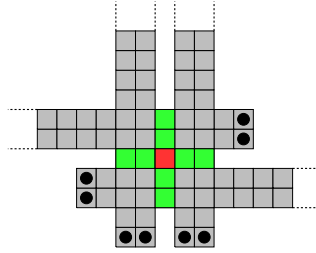
We give the details for the part in Fig. 11; the adaption to Fig. 12 is straightforward. The used arguments are highlighted in Fig. 15. First, there are already 12 corner pixels, so we have only 6 turns left to cover this local part. Note that we assume that we can appropriately place turns outside this local part, so we are using only local arguments and the turn constraints remain feasible in the global construction. We need at least 6 additional turns: one on each yellow area due to Lemma 2 and two on the green area due to Lemma 3. There can be no turn on the red pixel; otherwise we would need three of the six turns on the green and red pixels, contradicting the fact that we already need at least four in the yellow locations. In order to remain locally optimal, turns on the gray and red pixel thus are impossible; however, these pixels still need to be covered, in particular the pixels marked by hollow circles. Lemma 3 applied on these pixel forces us to place a turn on the adjacent yellow pixel above or below them, leaving only two further turns for covering the red pixel. Lemma 2 additionally forbids turns on two pixels to the left and right of the red pixel: We need the ability to make turns outside in order to fulfill it. ◀

► **Lemma 5.** *Given a construction, as described above, for an arbitrary formula with v variables and c clauses. Let k be the number of corner pixels in the zig-zagging paths of the variable gadgets (highlighted in blue). Then every cycle cover has to have at least $20v + 42c + k$ turns.*

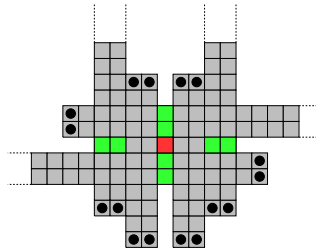
Proof. There is no interference between these local bounds, because in the proof we did not charge for exterior (not inside considered local part) turns made by possible cycles. Thus, the claim follows by summing over the local bounds for all local parts using Lemma 4. ◀

2.1.4 An Upper Bound for Satisfiable Formulas

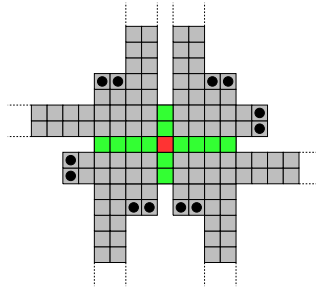
We can construct a solution that matches the lower bound given in Lemma 5 if the corresponding formula of the grid-graph construction is satisfiable.



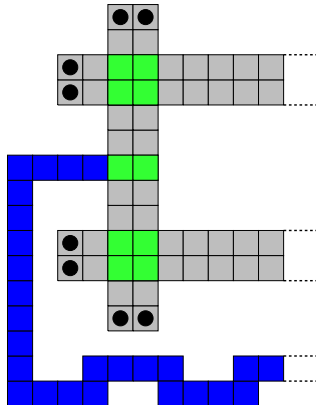
■ **Figure 10** 8 unique turns and 2 turns in green area



■ **Figure 11** 16 unique turns and 2 turns in green area

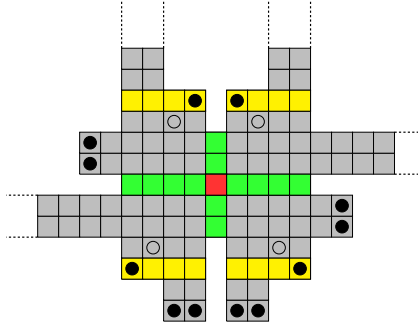


■ **Figure 12** 12 unique turns and 2 turns in green area



■ **Figure 13** 8 unique turns and 2 turns in green area.

■ **Figure 14** Lower bounds on the number of turns in the different parts of the construction (possibly reflected). The black dots represent turn position necessary to meet the lower bound.



■ **Figure 15** Auxiliary figure for the proof of Lemma 4.

► **Lemma 6.** *Given a construction for an arbitrary formula with v variables and c clauses, as described above. Let k be the number of corner pixels in the zig-zagging paths of the variable gadgets (highlighted in blue). If the formula is satisfiable, there exists a cycle cover with $20v + 42c + k$ turns.*

Proof. It is straightforward to deduce this from the description of the construction and Fig. 4, so we only sketch the details. Cover the gray pixels using only simple green 4-turn cycles on the explicit turns (black dots), as shown in Fig. 14. Select an arbitrary but fixed satisfiable solution for the formula. If a variable assignment is **true**, add a cycle that traverses all blue pixels of the variable and all upper crucial pixels of the variable with a minimum number of turns. This cycle has a turn at every blue corner pixel belonging to the variable, two turns per crucial pixel, and two further turns at each end to connect to the blue pixels. If a variable assignment is **false**, proceed analogously, but for the lower row of crucial pixels belonging to this variable. As the assignment is feasible, there are either none or exactly two remaining crucial red pixels per clause gadget. If there are two, they are vertically adjacent and we can cover them by a simple blue 4-turn cycle. As Lemma 4 leaves us with exactly 2 turns to select per crucial pixel, this matches the lower bound. ◀

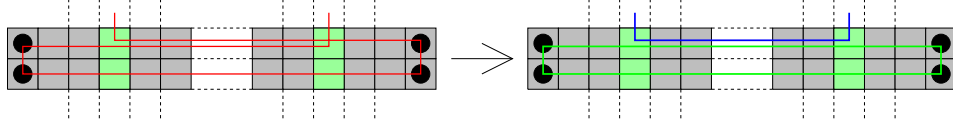
2.1.5 Lower Bound for Unsatisfiable Formulas

If the formula corresponding to the grid graph construction is not satisfiable, we can deduce that the number of turns given by Lemma 5 is not sufficient for a full coverage. We first show that in a cycle cover that matches the lower bound, we can separate a cycle cover for the gray pixels from the rest.

► **Lemma 7.** *Given an optimal cycle cover that matches the lower bound of Lemma 5, then we can modify the solution (without increasing the cost) such that it contains a cycle cover of exactly the gray pixel (the green cycles in Fig. 4).*

Proof. The cycle cover matches the lower bound of Lemma 5 and hence has to fulfill the restriction on the position of turns as in Lemma 4. We not only know that there has to be a turn at the exact same locations as the green cycles in Fig. 4, marked by black dots in Fig. 14, but also that these turns have to be exactly the same simply due to a lack of alternatives: if there are only potential partner turns in two directions, we can only make a turn between these two (u-turns are already prohibited by Lemma 5). If two such partnered turns are not directly connected, we know that there are exactly two turns on pixel that are highlighted in green in Fig. 14 in between them. We can simply connect these two turns

on green pixels directly as well as the two turns on black dots, as shown in Fig. 16. If the lower bound is matched, we can therefore always separate the green cycles as in Fig. 4.



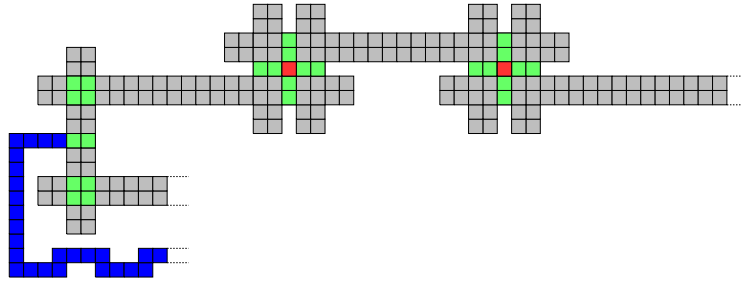
■ **Figure 16** Lemma 2 enforces the turns on the green pixels to be at the same y -coordinates. The cycle part can easily be separated to a cycle that covers the gray pixels and a connection between the turns in the green pixels.

◀

This allows us to only concentrate on covering the crucial and the blue pixels, because all gray pixels are already covered by the “black dot” cycles. We now show that every cycle cover that matches the lower bound results in a variable assignment (the cycle that covers the blue pixels either selects the **true** line or the **false** line and does not switch in between).

► **Lemma 8.** *In a cycle cover that matches the lower bound of Lemma 5, the cycle that covers the blue pixels of a variable also covers either all crucial pixels belonging to a **true** assignment of this variable and no other crucial pixel or all crucial pixels belonging to a **false** assignment of this variable and no other crucial pixel.*

Proof. It is impossible to cover a blue pixel twice without exceeding the lower bound, because we only have free turns in the green pixels. Therefore, the “variable” cycle has to go through the gray and crucial pixels. We are only allowed to make turns in the green pixels as marked in Fig. 14 and only at most one in every connected green area. The variable cycle can take the upper or the lower path (Fig. 13), but cannot switch between them, as there is only one possible “partner” turn to achieve an even number of turns on every strip. The only possible choice for a partner turn is the next turn of the cycle after applying Lemma 7. This forces the path to continue until it closes the cycle. See Fig. 17 for an illustration. ◀



■ **Figure 17** If the cycle uses the upper path, Lemma 2 forces us to cover exactly all crucial pixels along this way, as we have only one choice for the position of the “partner” turn.

Now only the crucial pixels not covered by the variable assignment cycles are left. We have only two turns for each of them, hence we need at least two still uncovered crucial pixels in any additional cycle; this cycle must have at most twice the number of turns than the number of its newly covered crucial pixels. This, however, is impossible for unsatisfied clauses, so we must exceed the lower bound.

► **Lemma 9.** *Given a 2-dimensional grid graph produced by the above procedure for an arbitrary One-in-three 3SAT formula. Let \mathcal{C} be a set of cycles with minimum turn costs. Let v be the number of variables in the corresponding formula, c the number of clauses, and k be the number of blue corner pixels. If the formula is not satisfiable, the cycle cover has to have at least $20v + 42c + k + 1$ turns.*

Proof. Assume there exists a cycle cover with only $20v + 42c + k$ turns (matching the lower bound of Lemma 5). We now show that this number of turns cannot suffice to cover all pixels. By separating the cycles on the black dotted pixels using Lemma 7, only $4v + 18c + k$ turns remain for the red and blue pixels.

Due to Lemma 8 we have to cover one line of crucial pixels per variable by a blue variable assignment cycle. The remaining crucial pixels can only be covered by the red cycles known from the construction. However, the use of red cycles in a way that no crucial pixel is covered multiple times (which would exceed the budget) enforces the logic described in the construction. A valid selection of red cycles is therefore only possible if the corresponding variable assignment is feasible. Because this is not the case, $20v + 42c + k$ turns cannot be sufficient. ◀

2.1.6 NP-Hardness

Finally, we can state the main theorem: the bound on the number of turns can only be met if the formula is satisfiable.

► **Theorem 10.** *Given a construction, as described above, for an arbitrary formula with v variables and c clauses. Let k be the number of blue highlighted corner pixels. Then the formula is satisfiable if and only if there exists a cycle cover with $20v + 42c + k$ turns.*

Proof. This follows from Lemma 6 and Lemma 9. ◀

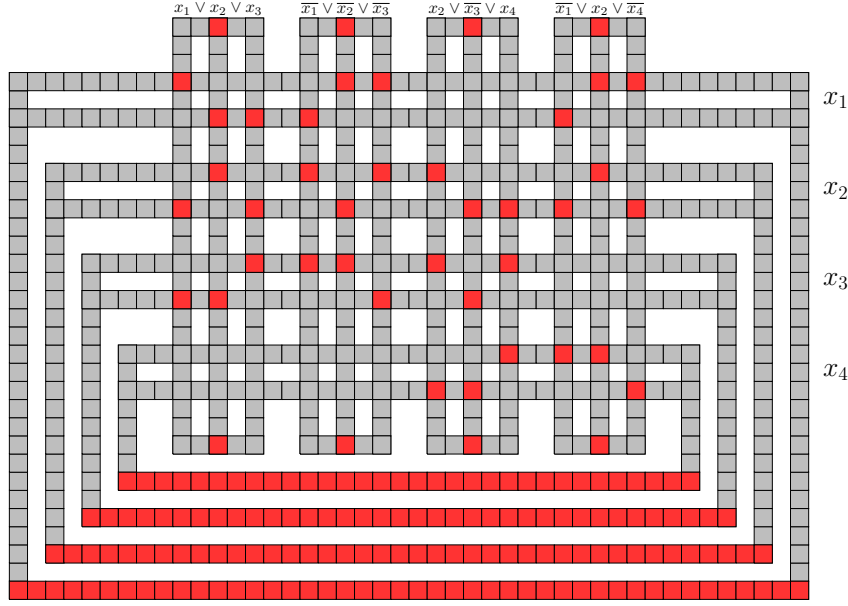
This concludes the proof of the NP-hardness of full coverage cycle cover (Theorem 1).

2.2 Subset Coverage in Thin Grid Graphs

For subset cover we can also show hardness for *thin* grid graphs. Arkin et al. [5, 6] exploits the structure of these graphs to compute an optimal minimum-turn cycle cover in polynomial time. If we only have to cover a subset of the vertices, the problem becomes NP-hard again. The proof is inspired by the construction of Aggarwal et al. [3] for the angular-metric cycle cover problem and significantly simpler than the one for full coverage.

► **Theorem 11.** *The minimum-turn subset cycle cover problem is NP-hard, even in thin grid graphs.*

Proof. The proof is inspired by the construction of Aggarwal et al. [3] for the angular-metric cycle cover. See Fig. 18 for an overview of the construction; due to its relative simplicity, we only sketch the details. Every variable consists of a U-shape, with two rows connecting the ends of the U. The bottom of the U has to be covered in any case, so we are free to either cover the upper or the lower row without additional turn cost. Covering the upper row means setting the variable to **true**, covering the lower row to **false**. These two rows intersect with the constructions for the clauses. Every clause construction consists of three vertical columns (intersecting with the variable constructions) and a horizontal connection at each end. At the top and the bottom of each is a pixel that has to be covered. The cheapest way to cover these two pixels is by a cycle that goes through the columns. Thus, we are able



■ **Figure 18** Example for the NP-hardness reduction using the *One-in-three 3SAT* formula $(x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_2 \vee \overline{x_3} \vee x_4) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_4})$.

to cover two of three columns per clause construction for free. The two covered columns represent the two literals of the clause that have to be **false**.

We mark additional pixels at the intersections of the variable constructions and the clause constructions, so that these pixels must also be covered. This is done in a such way that if we set a variable to a specific value for all clauses that now become **true**, the two columns to be covered are automatically enforced. In the example, if we set x_1 to **true** and cover the upper row, we have to use the two lines on the right in clause $x_1 \vee x_2 \vee x_3$, because otherwise the two lower pixels in the intersection are not covered. On the other hand, this enforces x_2 and x_3 to be **false**, as the corresponding two left pixels to be covered can no longer be covered by the clause's cycle. The cycle cover represents a valid solution for the *One-in-three 3SAT* formula, if and only if we do not need any additional cycles. ◀

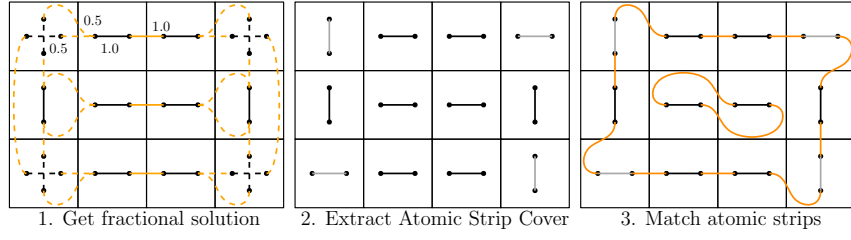
Arkin et al. [5, 6] showed how the structure of thin grid graphs can be exploited for computing an optimal minimum-turn cycle cover in polynomial time. If we only have to cover a subset of the vertices, the problem is NP-complete.

3 Approximation Algorithms

3.1 Cycle Cover

Now we describe a 2ω -approximation algorithm for the semi-quadratic (penalty) cycle cover problem with ω atomic strips per point if the edge weights satisfy Eq. (1). We focus on the full coverage version, as the penalty variant can be modelled in full coverage (with equal ω and while still satisfying Eq. (1)), by adding for every point $p \in P$ two further points that have a zero cost cycle only including themselves and a cycle that also includes p with the cost of the penalty.

Our approximation algorithm proceeds as follows. We first determine an atomic strip cover via linear programming. Computing an optimal atomic strip cover is NP-hard; we can



■ **Figure 19** Example of the approximation algorithm for a simple full cycle cover instance in a grid graph. First the fractional solution of the integer program (2)-(4) is computed. Strips and edges with value 0 are omitted, while dashed ones have value 0.5. Then the dominant (i.e., highest valued) atomic strips of this solution are selected. Finally, a minimum weight perfect matching on the ends of the atomic strips is computed. (Recall that atomic strips only have an but no length, so the curves in the corner indicate simple 90° turns.)

show that choosing the *dominant* strips for each pixel in the fractional solution, i.e., those with the highest value, suffices to obtain provable good solutions. As a next step, we connect the atomic strips to a cycle cover, using a minimum-weight perfect matching. See Fig. 19 for an illustration.

We now describe the integer program whose linear programming relaxation is solved to select the dominant atomic strips. It searches for an optimal atomic strip cover that yields a perfect matching of minimum weight. To satisfy Eq. (1), transitive edges may need to be added, especially loop-edges (which are not used in the final solution). The IP does not explicitly enforce cycles to contain at least two points: all small cycles consist only of transitive edges that implicitly contain at least one further atomic strip/point. For the usage of a matching edge $e = vw \in E_O$, we use the Boolean variable $x_e = x_{vw}$. For the usage of an atomic strip $o = vw \in O_p, p \in P$, we use the Boolean variable $y_o = y_{vw}$.

$$\min \sum_{e \in E_O} \text{cost}(e) x_e \quad (2)$$

$$\text{s.t.} \quad \sum_{vw \in O_p} y_{vw} = 1 \quad p \in P \quad (3)$$

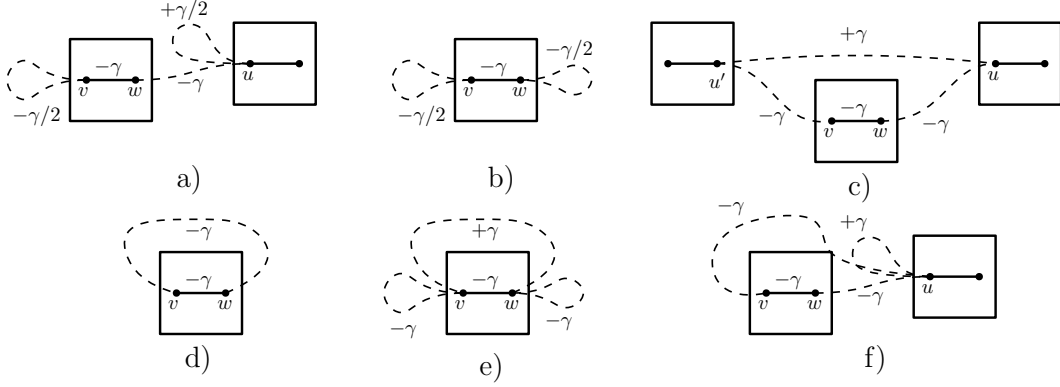
$$2x_{vv} + \sum_{\substack{e \in E_O(v) \\ e \neq vv}} x_e = 2x_{ww} + \sum_{\substack{e \in E_O(w) \\ e \neq ww}} x_e = y_{vw} \quad p \in P, vw \in O_p \quad (4)$$

We minimize the cost of the used edges, with Eq. (3) forcing the selection of one atomic strip per pixel (atomic strip cover) and Eq. (4) ensuring that exactly the vertices (endpoints) of the selected atomic strips are matched, with loop edges counting double due to their two ends.

► **Theorem 12.** *Assuming edge weights that satisfy Eq. (1), there is a 2ω -approximation for semi-quadratic (penalty) cycle cover.*

Proof. We prove this theorem by showing that there exists a matching on the dominant atomic strips of the fractional solution with at most 2ω the cost of the fractional solution. This implies that the minimum weight perfect matching on these strips has to be at least that good.

Due to the constraints of type Eq. (3), the dominant strip $o \in O_p$ for every $p \in P$ has a value of $y_o \geq 1/\omega$ in any feasible solution of the LP-relaxation. Let o_p be this dominant strip for $p \in P$ and $V'_O \subseteq V_O$ the set of the endpoints of all of them. There may also be further $p \in P, o \in O_p$ with $y_o > 0$. We can set these to zero using the operations of Fig. 20 without



■ **Figure 20** Operations for reducing the usage of a strip vw . Strips are shown by thick lines, while edges are indicated by dashed lines. These costs do not increase (for $\gamma \geq 0$), due to Ineq. 1. For some cases as a) one needs a repetitive application of this inequality: first replace the $\gamma/2 * vv$ loop-edge by $\gamma/2 * vu$ while reducing uw by $\gamma/2$ and then replace the $\gamma/2 * wu$ and the $\gamma/2 * vu$ by $\gamma/2 * uu$. As long as the usage is greater than zero, at least one of the operations is possible.

increasing the objective value and without changing any y_{o_p} while also satisfying Eq. (4), because of the constraint in Eq. (1). The resulting solution only violates Eq. (3) by relaxing it to $\sum_{vw \in O_p} y_{vw} \geq 1/\omega$, but we fix this later by a simple multiplication. Now we only have edges from $E'_O = E_O \cap (V'_O \times V'_O)$ in our fractional solution. The corresponding objective value is still a lower bound on the optimal solution of the integer program.

We can multiply the solution by ω resulting in $y_{o_p} \geq 1$ for all $p \in P$. We can apply the same procedure to reduce all $y_{o_p} > 1$ to = 1. The new solution has a cost of at most ω times the optimal solution and all variables $y_o, o \in O_p, p \in P$ are now Boolean; however, the edge usages can still be fractional.

If we now fix the values for $y_o, o \in O_p, p \in P$ and remove all vertices (and their incident edges) that are not in V'_O , we have a minimum perfect matching polytope with loop-edges as stated below.

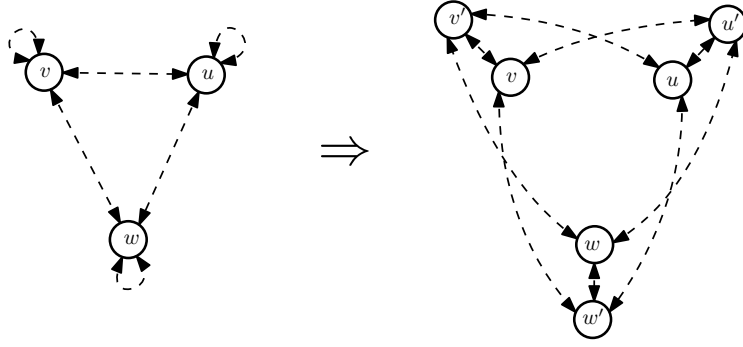
$$\min \quad \sum_{e \in E'_O} c(e) * x_e \quad (5)$$

$$\text{s.t.} \quad \sum_{e=vw \in E'_O} x_e + 2 * x_{vv} = 1 \quad \forall v \in V'_O \quad (6)$$

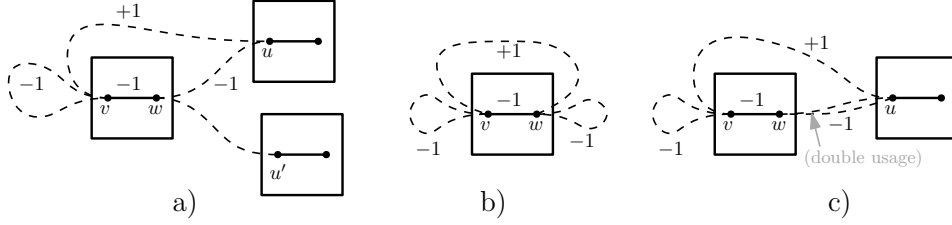
$$0 \leq x_e \leq 1 \quad \forall e \in E'_O \quad (7)$$

Because the previous solution is still feasible in this formulation, the optimal solution of this formulation is also at most ω the costs of the optimal solution of the problem.

We can show that the new polytope contains a half-integral optimal solution. The proof works analogously to the proof of the half-integrality of the classic matching polytope (see, e.g., Theorem 6.13 in the book of Cook et al. [14]). We create a bipartite graph $G'_{\text{bip}}(V'_O \times 2, E'_{\text{bip}})$ in which for every $v \in V'_O$ there are two vertices $v, v' \in V'_O \times 2$ and for every edge $e = vw \in E'_O$ there are the edges vw' and $v'w$ in E'_{bip} , see Fig. 21. A loop-edge $vv \in E'_O$ is simply replaced by a single edge $vv' \in E'_{\text{bip}}$. Because there are no edges within the individual copies of V'_O in G'_{bip} , the graph is bipartite; the bipartite matching polytope is known to be integral. By assigning the edges in G'_{bip} the same values as for the solution on $G'_O(V'_O, E'_O)$ (doubling for loop-edges), we obtain a feasible solution with twice the costs on G'_{bip} . The optimal solution in G'_{bip} , hence, at most twice as expensive as the optimal solution in G'_O . The optimal solution in G'_{bip} (which is integral) can be transformed to a solution of half the costs for G'_O by assigning each edge half the sum of the corresponding



■ **Figure 21** Converting a graph to a bipartite graph. Every edge is doubled except the loop-edges which become regular edges. The matching vw', wu', uv' would be converted to the matching $0.5vw, 0.5wu, 0.5uv$. There always exists a perfect matching in the original graph for our problem.



■ **Figure 22** Removing integral loop-edges. A loop-edge always implies a double usage of the corresponding strip. The costs do not increase due to Ineq 1. A ‘+1’ indicates a new edge, a ‘-1’ the removal of an edge.

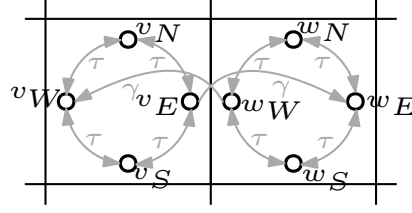
two edges (or one edge for loop-edges) in the solution for G'_{bip} . This solution is half-integral and optimal. Hence, we have an optimal solution where regular edges are selected by 0, 0.5, or 1 and loop-edges are selected by 0 or 0.5.

If we double such a solution, we obtain an integral solution with at most $2 * \omega$ the costs of the original linear program. Every vertex has either two regular edges selected with 1 each, a single regular edge selected twice, or a loop-edge selected with 1. We can remove the loop-edges without cost increase as shown in Fig. 22 by also reducing the strip usage to one (a loop-edge always implies a double usage). For removing other double usages we can use nearly the same technique as for fractional in Fig. 20. We only have to make sure never to create fractional loop-edges (we do not need the cases in which this can happen because these cases all have loop-edges in the beginning which we removed before). Integral loop-edges can immediately be removed as in Fig. 22.

In the end, we have a solution that is a matching for G'_O and has at most 2ω the cost of the objective value of the linear relaxation. As the linear relaxation provides a natural lower bound, this concludes the proof. ◀

3.2 Tours

A given cycle cover approximation can be turned into a tour approximation at the expense of an additional constant factor. Because every cycle involves at least two points and a full rotation, we can use classic tree techniques known for TSP variants to connect the cycles and



■ **Figure 23** We can easily compute the shortest path with turn costs in grid graphs by replacing every point (that only encodes a position) by vertices for all headings of interest (four for grid graphs). The turn costs (τ) then are encoded in the costs of switching between these vertices. A vertex heading north allows only to go north such that the rotation edges have to be used to make a rotation. γ denotes the distance cost.

charge the necessary turns to the involved cycles. We sketch the basic ideas and elaborate the details for the most difficult case afterwards in Theorem 14 (the other cases are analogous).

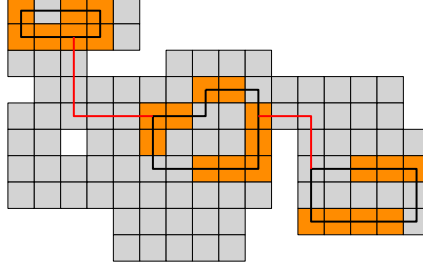
For the classic Traveling Salesman Problem with triangle inequality, minimum spanning trees are trivial lower bounds, as any tour must contain a spanning tree. This is also true for the penalty TSP and the prize-collecting Steiner tree; note that “penalty” and “prize-collecting” variants are completely equivalent. Doubling optimal trees yields trivial 2-approximations. (The prize-collecting Steiner tree is NP-hard, but there is a 2-approximation [22].)

This is not directly possible with turn costs, because it matters from where a vertex is entered. However, if we already have a cycle cover and we aim to connect them, this gets significantly easier. If there is a path between two cycles, we can double it and merge the cycles, requiring not more than an additional 180° turns at the ends of the paths, regardless of the direction from which the path hits the cycle. Thus we can merge two cycles with a cost of two times the cheapest path between them, plus 360° for connecting the elements. The costs of the paths can be charged to the optimal tour, analogous to the classic TSP. On the other hand, the 360° can be charged to one of the cycles, because every cycle needs to do at least a full rotation, and there are fewer such merge processes than cycles in the initial cycle cover. This directly yields a method for approximating subset tours when a cycle cover approximation is given, see Fig. 24.

In order to connect the cycles of a penalty cycle cover to a penalty tour, we cannot simply use the doubled minimum spanning tree technique like we did for subset coverage; e.g., if the penalty cycle cover consists of two distant cycles, it may be cheaper not to connect them, but select the better cycle as a tour and discard the other cycle. Instead, we double a prize-collecting Steiner tree instead of a minimum spanning tree, in which every vertex not in the tree results in a penalty (or every vertex in the tree provides a prize). Without loss of generality, we may assume that all cycles are disjoint: otherwise, we connect two crossing cycles at a cost of at most 360° , which can be charged to the merged cycle. Then the penalty for not including a cycle is the sum of penalties of all its pixels. As a result, we get a constant-factor approximation for penalty tours.

► **Theorem 13.** *Assuming validity of Eq. 1 we can establish the following approximation factors for tours.*

1. *Full tours in regular grid graphs: 6-approximation.*
2. *Full tours in generalized grid graphs: 4ω -approximation.*
3. *Subset tours in (generalized) grid graphs: $(4\omega + 2)$ -approximation.*
4. *Geometric full tours: $(4\omega + 2)$ -approximation.*
5. *Penalty tours (in grid graphs and geometric): $(4\omega + 4)$ -approximation.*



■ **Figure 24** Connecting subset cycles (cycles in black, subset pixel in orange) by a minimum spanning tree (red edges) on the components/cycles.

These results also hold for objective functions that are linear combinations of length and turn costs.

Proof. Crucial are that (1) a cycle always has a turn cost of at least 360° , (2) two intersecting cycles can be merged with a cost of at most 360° , and (3) two cycles intersecting on a 180° turn can be merged without additional cost.

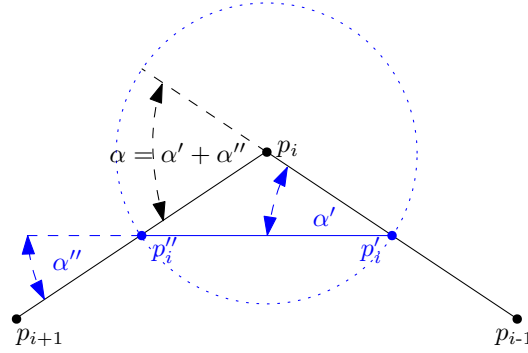
1. For full tours in grid graphs, greedily connecting cycles provides a tour with at most 1.5 times the turn cost of the cycle cover, while a local optimization can be exploited to limit the length to 4 times the optimum, as shown by Arkin et al. [5].
2. In a cycle cover for (generalized) grid graphs, there are always at least two cycles with a distance of one, while every cycle has a length of at least 2; otherwise the cycle cover is already a tour. This allows iteratively merging cycles at cost at most as much as a cheapest cycle; the total number of merges is less than the number of cycles.
3. and (iv) For subset coverage in grid graphs or full coverage in the geometric case, we need to compute the cheapest paths between any two cycles, ignoring the orientations at the ends. First connect all intersecting cycles, charging the cost on the vanishing cycles. The minimum spanning tree on these edges is a lower bound on the cost of the tour. Doubling the MST connects all cycles with the cost of twice the MST, the cost of the cycle cover, and the turn costs at the end of the MST edges, which can be charged to the cycles.
5. Penalty tours can be approximated in a similar manner. Instead of an MST, we use a Price-Collecting Steiner Tree, which is a lower bound on an optimal penalty tour. We use a 2-approximation for the PCST [22], as it is NP-hard. We achieve a cost of twice the 2-approximation of the PCST, the cost of the penalty cycle cover, and the cost of its cycles again for charging the connection costs. The penalties of the points not in the cycle cover are already paid by the penalty cycle cover.

◀

For completeness, we describe the details for 2-dimensional grid graphs; other cases are analogous.

► **Theorem 14.** *There is a 12-approximation algorithm for penalty tours in grid graphs.*

Proof. The factor of 12 results of 4OPT for a penalty cycle cover, $2 \cdot 2\text{OPT}$ for a 2-approximation of a prize-collecting Steiner tree, and again the cost of the penalty cycle cover as an upper bound on the necessary additional turns. Just like for full coverage, the prize-collecting Steiner tree is computed on an auxiliary graph based on the cycles. For simplicity, we directly merge all cycles that share a pixel. The cost for this can be charged to one of the cycles from the same budget as for the later connections without any interference,



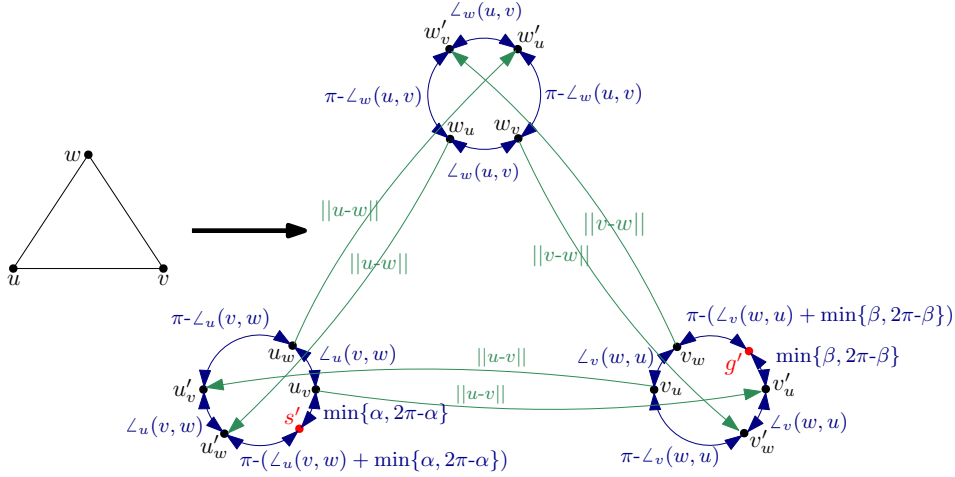
■ **Figure 25** If the turning point p_i is not on a vertex of an obstacle, we can build a shorter tour with the same turn costs by replacing p_i by p'_i and p''_i (blue).

because we reduce the number of cycles used in the later part. In the corresponding graph, every cycle is represented by a vertex. The penalty for points not covered by cycles is already paid for by the penalty cycle cover. There is an edge between any two cycles; its cost is the cheapest transition from one pixel of the first cycle to a pixel of the second cycle, ignoring the orientations at these end pixels. The penalty of each vertex is the sum of the penalties of all of its pixels. Doubling a prize-collecting Steiner tree and removing all cycles not in the tree results in a tour; the turns at the ends can be charged to the cycle cover, just like for full/subset coverage.

It remains to be shown that the optimal prize-collecting Steiner tree on the cycle graph is a lower bound on the optimal tour. This can be seen by obtaining such a tree from the optimal penalty tour. Because the turns at the end points are free, as is moving within the cycles, the resulting tree is not more expensive than the tour. Further, visiting a single pixel of a cycle covers also all other pixels of the cycle, so also the penalties are cheaper. Finally, a 2-approximation for the prize-collecting Steiner tree can be provided by the algorithm of Goemans and Williamson [22]. ◀

3.3 Geometric Instances

In this section we give some additional information on solving the 2-dimensional geometric instances with polygonal obstacles. We consider an (individual) set of ω atomic strips per point $p \in P$ and search for a cycle cover/tour that covers all points of P (by integrating one of the atomic strips) with minimal distance and turn angle sum. If there are only at most ω orientations per point, there are at most $(2\omega|P|)^2$ many possible transitions in the tour. The cheapest transition between two configurations can be computed by considering the visibility graph (this is analogous to the situation for Euclidean shortest paths, see Fig. 25 and e.g., Chapter 15 in de Berg et al. [16]) and a simple graph transformation to integrate the turn costs into edge costs, which requires $O(|E|)$ vertices and $O(|E|)$ edges, see Fig. 26. The visibility graph can be computed in $O(n^2)$, e.g., by the algorithm of Asano et al. [8]. Skipping an atomic strip does not increase the costs, hence we can use the Atomic Strip Matching technique described before. The connecting strategies remain also the same; the only additional subroutine is the computation of the geometric primitives, which remains polynomial.



■ **Figure 26** The transformation of the graph to integrate turn costs in the edge costs. The start configuration is in u heading south-east (s') and the goal configuration is in v heading north-east (g'). $\angle_a(b, c)$ denotes the counterclockwise angle from segment (a, b) to segment (a, c) .

4 Conclusions

We have presented a number of theoretical results on finding optimal tours and cycle covers with turn costs. In addition to resolving the long-standing open problem of complexity, we provided a generic framework to solve geometric (penalty) cycle cover and tours problems with turn costs.

As described in [11], the underlying problem is also of practical relevance. As it turns out, our approach does not only yield polynomial-time approximation algorithms; enhanced by an array of algorithm engineering techniques, they can be employed for actually computing optimal and near-optimal solutions for instances of considerable size in grid graphs. Further details on these algorithm engineering aspects will be provided in our forthcoming paper [19].

References

- 1 Pankaj K. Agarwal, Therese C. Biedl, Sylvain Lazard, Steve Robbins, Subhash Suri, and Sue Whitesides. Curvature-constrained shortest paths in a convex polygon. *SIAM J. Comp.*, 31(6):1814–1851, 2002.
- 2 Pankaj K. Agarwal and Hongyan Wang. Approximation algorithms for curvature-constrained shortest paths. *SIAM J. Comp.*, 30(6):1739–1772, 2000.
- 3 Alok Aggarwal, Don Coppersmith, Sanjeev Khanna, Rajeev Motwani, and Baruch Schieber. The angular-metric traveling salesman problem. *SIAM J. Comp.*, 29(3):697–711, 1999.
- 4 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.
- 5 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 *Proc. 12th ACM-SIAM Symp. Disc. Alg. (SODA)*, pages 138–147, 2001.
- 6 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 J. Comp.*, 35(3):531–566, 2005.

- 7 Esther M. Arkin, Sándor P. Fekete, and Joseph S. B. Mitchell. Approximation algorithms for lawn mowing and milling. *Comp. Geom.*, 17(1-2):25–50, 2000.
- 8 Takao Asano, Tetsuo Asano, Leonidas J. Guibas, John Hershberger, and Hiroshi Imai. Visibility of disjoint polygons. *Algorithmica*, 1(1):49–63, 1986.
- 9 Sebastian Astroza, Priyadarshan N Patil, Katherine I Smith, and Chandra R Bhat. Transportation planning to accommodate needs of wind energy projects. In *Transportation Research Board–Annual Meeting*, 2017. article 17-05309.
- 10 Giorgio Ausiello, Vincenzo Bonifaci, Stefano Leonardi, and Alberto Marchetti-Spaccamela. Prize-collecting traveling salesman and related problems. In Teofilo F. Gonzalez, editor, *Handbook of Approximation Algorithms and Metaheuristics*. Chapman and Hall/CRC, 2007.
- 11 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 *Proc. 33rd Symp. Comp. Geom. (SoCG)*, pages 62:1–62:5, 2017. Video at <https://www.youtube.com/watch?v=SFyOMDgdNao>.
- 12 Nadia M. Benbernou. *Geometric Algorithms for Reconfigurable Structures*. Ph.D. thesis, Massachusetts Institute of Technology, 2011.
- 13 Jean-Daniel Boissonnat and Sylvain Lazard. A polynomial-time algorithm for computing a shortest path of bounded curvature amidst moderate obstacles. In *Proc. 12th Symp. Comp. Geom. (SoCG)*, pages 242–251, 1996.
- 14 W.J. Cook, W.H. Cunningham, W.R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. Wiley-Interscience, 1997.
- 15 Igor R. de Assis and Cid C. de Souza. Experimental evaluation of algorithms for the orthogonal milling problem with turn costs. In *Proc. 10th Symp. Exp. Alg. (SEA)*, pages 304–314, 2011.
- 16 Mark De Berg, Otfried Cheong, Marc Van Kreveld, and Mark Overmars. *Computational Geometry*. Springer, 2008.
- 17 Erik D. Demaine, Joseph S. B. Mitchell, and O’Rourke Joseph. The Open Problems Project. <http://cs.smith.edu/orourke/TOPP/>.
- 18 Lester E Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American J. Math.*, 79(3):497–516, 1957.
- 19 Sándor P. Fekete and Dominik Krupke. Practical methods for computing large covering tours and cycle covers with turn cost. In *Proc. 21st SIAM Workshop Alg. Engin. Exp. (ALENEX)*, 2019. To appear.
- 20 Sándor P. Fekete and Gerhard J. Woeginger. Angle-restricted tours in the plane. *Comp. Geom.*, 8:195–218, 1997.
- 21 Mike Fellows, Panos Giannopoulos, Christian Knauer, Christophe Paul, Frances A. Rosamond, Sue Whitesides, and Nathan Yu. Milling a graph with turn costs: A parameterized complexity perspective. In *Proc 36th Worksh. Graph Theo. Conc. Comp. Sci. (WG)*, pages 123–134, 2010.
- 22 Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comp.*, 24(2):296–317, 1995.
- 23 Sylvain Lazard, John Reif, and Hongyan Wang. The complexity of the two dimensional curvature-constrained shortest-path problem. In *Proc. 3rd Worksh. Alg. Found. Robotics (WAFR)*, pages 49–57, 1998.
- 24 Olaf Maurer. Winkelminimierung bei Überdeckungsproblemen in Graphen. Diplomarbeit, Technische Universität Berlin, 2009.
- 25 Jerome Le Ny, Eric Feron, and Emilio Frazzoli. On the Dubins traveling salesman problem. *IEEE Trans. Autom. Contr.*, 57(1):265–270, 2012.

- 26 Thomas J. Schaefer. The complexity of satisfiability problems. In *Proc. 10th ACM Symp. Theo. Comput. (STOC)*, pages 216–226, 1978.
- 27 Clifford Stein and David P. Wagner. Approximation algorithms for the minimum bends traveling salesman problem. In *Proc. 8th Int. Conf. Int. Prog. Comb. Opt. (IPCO)*, pages 406–422, 2001.
- 28 Ryo Takei, Richard Tsai, Haochong Shen, and Yanina Landa. A practical path-planning algorithm for a simple car: A Hamilton-Jacobi approach. In *Proc. 29th American Cont. Conf. (ACC)*, pages 6175–6180, 2010.