The Computational Complexity of Finding Hamiltonian Cycles in Grid Graphs of Semiregular Tessellations

Kaiying Hou* Jayson Lynch†

Abstract

Finding Hamitonian cycles in square grid graphs is a well studied and important question. Recent work has extended these results to triangular and hexagonal grids, as well as further restricted versions such as solid or thin grids [7, 8, 4]. In this paper, we examine a class of more complex grids, as well as investigate the problem with restricted types of paths. We investigate the hardness of Hamiltonian cycle problem in grid graphs of semiregular tessellations. We give NP-hardness reductions for finding Hamiltonian paths in grid graphs based on all eight of the semiregular tessellations. Next, we investigate variations on the problem of finding Hamiltonian Paths in grid graphs when the path is forced to turn at every vertex. Related problems were considered in[6]. We show deciding if 3D square grid graphs admit a Hamiltonian cycle is NP-complete, even if the height of the grid is restricted to 2 vertices. We give a polynomial time algorithm for deciding if a solid square grid graph admits a Hamiltonian cycle which visits vertices at most twice and turns at every vertex.

1 Introduction

The Hamiltonian cycle problem (HCP) in grid graphs has been well studied and has led to application in numerous NP-hardness proofs for problems such as the milling problem [2], Pac-Man [10], finding optimal solutions to a Rubik's Cube [3], and routing in wireless mesh networks [11]. The problem has been of interest to computer scientists for many years and recently a number of variations on the problem have been investigated. Itai, Papadimitriou, and Szwarcfiter proved that the HCP in square grid is NP-complete by reducing from the HCP in planar max-degree-3 bipartite graphs [7]. More recently, the HCPs in triangular and hexagonal grid were shown to be NP-complete[8]. This paper also introduced several new constrains on grid graphs, such as being thin or polygonal. Several of those open problems were solved by Demaine and Rudoy [4] by reducing from 6-Regular Tree-Residue Vertex Breaking problem (TRVB) [5]. These papers also show results on grid graphs with restrictions such as thin, polygonal, and solid. With all the interest in the computational complexity of the HCP in grid graphs, it is reasonable to ask whether we can generalize or adapt these results to different types of grids. In addition, we investigate the notion of angle-restricted tours, studied in [6], in the context of grid graphs. We give both algorithms and hardness proofs for finding Hamiltonian paths with this 'always-turning' constraint.

Although the hardness of the HCP in semiregular grids seems like an abstract question, it has many possible applications. Grids are natural structures that things may be formatted into. For example, the layout of buildings or modular structures used in space may form a network that follow the patterns of semiregular, or more general, grids. If certain locations in such networks need to be visited for maintenance, and one wants an optimal route, then this is well-modeled by the Hamiltonian path problem. Our reductions both give insight into what sorts of regular structures will be difficult to find optimal paths for, as well as ways of potentially transferring other efficient algorithms to these new problems. Finally, the results and techniques in this paper may be useful in proving hardness of other problems by reducing from HCPs in semiregular grids.

The always-turning Hamiltonian path problems also has some relation to more concrete questions. First, one can see the always-turning constraint as path planning in a world with reflections at fixed angles and locations. One may be routing optics to various locations on an optics table. Reflections of 45 degrees in a grid-based world are also a common element in puzzles and games. In addition, there has been study of problems which try to minimize the number of turns taken in a covering tour [1]. In many ways this can be seen as the opposite, modeling a case where turning is significantly easier than continuing straight.

A full version of this paper is available on the arXiv[1].

Results In Section 3 we extend the class of grid graphs studied to those based on semiregular tessellations. There are a total of eight semiregular tessellations [12], which are shown in Figure 1. For all eight semiregular tessellations we show the corresponding Hamiltonian cycle problem in the induced grid graph is NP-complete.

*Phillips Academy Andover, khou@andover.edu
†MIT Computer Science and Artificial Intelligence Laboratory, jaysonl@mit.edu

[1]arXiv:1805.03192

We show hardness by reducing from three NP-complete problems: HCP in planar max-degree-3 bipartite graphs [7], HCP in hexagonal grids [8], and TRVB [4].

In Section 4 we examine the question of Hamiltonian paths which turn at every vertex. We show this problem is hard in 3D square grid graphs. We also show it is easy in triangular grids with only 60 degree turns but hard in triangular grids when 120 degree turns are allowed. Finally, we examine a problem in square grids where a path must visit every vertex at least once, must turn at every vertex, and cannot reuse edges. We give a linear time algorithm for solving this double turning problem in solid square grid graphs.

2 Definitions

A *tessellation* is a tiling of a plane with polygons without overlapping. A *semiregular tessellation* is a tessellation which is formed by two or more kinds regular polygons of side length 1 and in which the corners of polygons are identically arranged. Figure 1 depicts part of each of the eight semiregular tessellations.

An infinite lattice of a semiregular tessellation is a lattice formed by taking the vertices of the regular polygons in the tessellation as the points of the lattice. A graph G is *induced* by the point set S if the vertices of G are the points in S and its edges connect vertices that are distance 1 apart. A *grid graph* of a semiregular tessellation, or a *semiregular grid*, is a graph induced by a subset of the infinite lattice formed by that tessellation. Call the infinite graph induced by the full lattice a *full grid*.

A *pixel* is the simple cycle bounding a face in a grid graph that contains the same bounding edges and vertices as the corresponding face in the full grid. Thus a pixel can be thought of as a cycle in a graph which bounds precisely one tile in the original tessellation. We may use pixel interchangeably to refer to the bounding cycle, the face bound by the cycle, or the set of vertices around that face. A *solid grid graph* is one in which every bounded face is a pixel.

A *Hamiltonian cycle* is a cycle that passes through each vertex of a graph exactly once. The *Hamiltonian cycle problem*, sometimes abbreviated as HCP, asks that given a graph, whether or not that graph admits a Hamiltonian cycle. The HCP in a semiregular tessellation asks, given a grid graph of that tessellation, whether it admits a Hamiltonian cycle.

3 Finding Hamiltonian Paths in Semi-Regular Tessellations is NP-Complete

This section shows the NP-completeness of HCPs in all eight semiregular tessellations. There are three NP-complete problems that we reduce from: the HCP in hexagonal grid, the HCP in planar max-degree-3 bipartite graphs, and the TRVB problem. We give some representative reductions and leave the rest of the constructions to Appendix A.

Theorem 1 *The HCP in grid graphs of the 3.4.6.4 tessellation is NP-complete.*

Proof. We will reduce from the HCP in hexagonal grids. Given a hexagonal grid graph G', we will construct a grid graph G of the 3.4.6.4. tessellation in this way: for every edge in G' we add the edge gadget shown in Figure 2 to G and for every vertex in G' we add the vertex gadgets shown in Figure 3 to G. Since the 3.4.6.4 tesselation has scaled versions of the translational symmetries of the hexagonal grid, picking an embedding for our construction is straightforward. An example can be seen in Figure 4. Since the hexagonal grid G' is bipartite, we can design different vertex gadgets for each end of the edges. We call the two classes of vertices even and odd vertices.

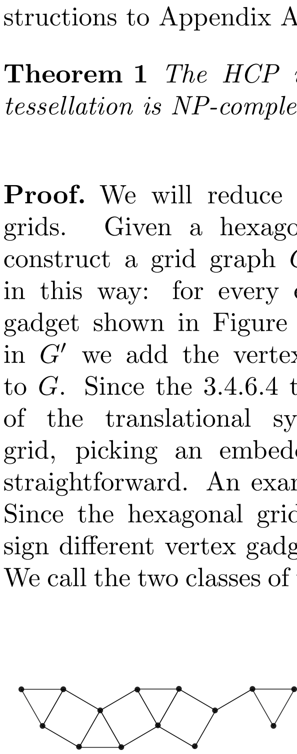

Figure 2: Edge gadget

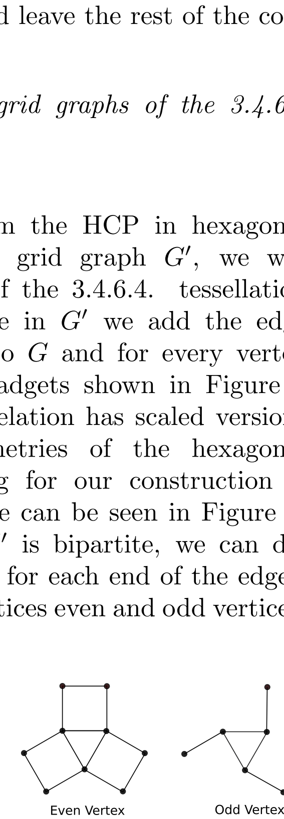

Figure 3: Vertex gadgets

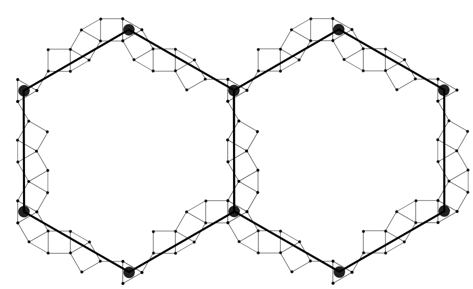

Figure 4: A simulated graph

Now, we will show that the original graph G' has a Hamiltonian cycle C' if and only if the simulated graph G has a Hamiltonian cycle C. If the G' has a Hamiltonian cycle C', for any taken edge in it, we go through the corresponded edge gadget in G with the cross path in Figure 5; for any untaken edge, we go through the corresponded edge gadget with the return path. Because the simulated vertices in G are triangles (K_3), there is always a path to take the simulated vertex by entering from one point and leaving at the other. Therefore, if there is a Hamiltonian cycle C' in the original graph G', then there is a Hamiltonian cycle

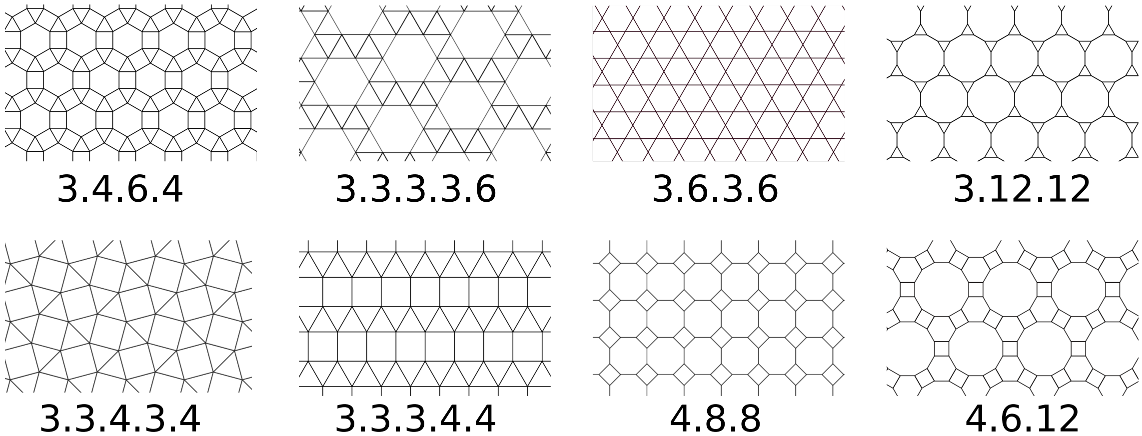

Figure 1: The eight semiregular tessellations. It is common to refer to them by the size of the faces while walking around a vertex.

C in the simulated graph G.

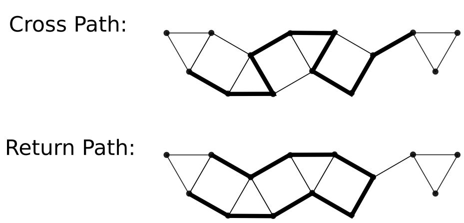

Figure 5: Two kinds of paths in the 3.4.6.4 edge gadget

The essential difference between the cross path and the return path is that a cross path starts and finishes at different ends of an edge while the return path starts and finishes in the same end. Note that the return and cross paths are the only two paths which go through the edge gadget and visit all of its vertices. The odd vertex gadget is connected to the edge gadget through a single edge connection which prevents the return path from entering the odd vertex gadget. If a Hamiltonian cycle C exists in the simulated graph G, each odd vertex gadget in G must be connected to two cross paths and the even vertex gadgets can either be connected to two cross paths or two cross paths and a return path. Then, we can find a cycle C' in the original graph G' by making each edge gadget with a cross path in C a taken edge in C'. Thus, if there is a cycle C in the simulated graph G, there is a cycle C' in the original graph G'. This way, we showed the original graph G' has a Hamiltonian cycle C' if and only if the simulated graph G has a Hamiltonian cycle C.

□

Theorem 2 *The HCP in grid graphs of the 3.3.3.3.6 tessellation is NP-complete.*

Proof. Similar to the 3.4.6.4 tessellation in Theorem 1, see Appendix A for details. □

Theorem 3 *The HCP in grid graphs of the 3.6.3.6 tessellation is NP-complete.*

Proof. Similar to the 3.4.6.4 tessellation in Theorem 1, see Appendix A for details. □

Theorem 4 *The HCP in grid graphs of the 3.12.12 tessellation is NP-complete.*

Proof. See Appendix A for the proof. □

Theorem 5 *The HCP in grid graphs of the 3.3.4.3.4 tessellation is NP-complete.*

Proof. We will reduce from HCP in planar max-degree-3 bipartite graphs. First observe that this tessellation can be viewed as a square grid with some extra diagonals. We directly use the gadgets of the square grid proof in the 1982 paper for constructing G [7]. The edge, even vertex and odd vertex gadgets are shown below. Note that these gadgets are identical to the square grid gadgets except they have some extra edges. In creating the simulated graph G based on a planar max-degree-3 bipartite graph G', we go through the same process as that in the square grid reduction: first create a parity-preserving embedding of the max-degree-3 bipartite graph; then replace the edges and vertices of the embedding with respective gadgets [7].

There are only two kinds of traversals for the edge gadget: cross paths and a return paths. Although there is more than one kind of cross path due to the extra edges, they have the essential characteristic of starting from one end of the gadgets and finishing at the other end (unlike the return path that begins and finishes at the same end). Another difference from the square grid reduction is that the odd vertex

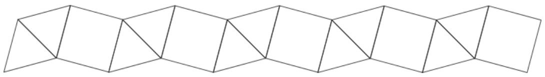

Figure 6: Edge gadget for 3.3.4.3.4

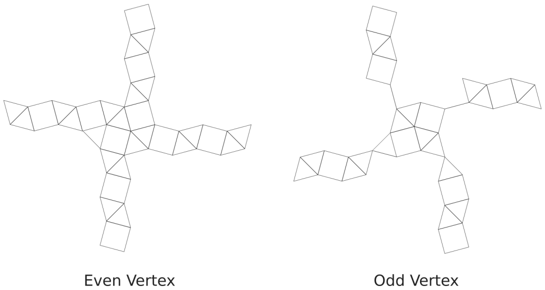

Figure 7: Vertex gadgets for 3.3.4.3.4

gadgets connect to the bottom edge gadget through a single point rather than a single edge as the other edge gadgets. This single point connection also prevents a return path from entering the odd vertex gadget. We call the single edge and single point connections that the path only enters and exits odd vertices once. Since the graph is bipartite, this forces the other two edges to be return paths, ensuring our simulated path can only enter and exit each vertex once. □

Theorem 6 *The HCP in grid graphs of the 3.3.3.4.4 tessellation is NP-complete.*

Proof. See Appendix A. □

We now show the HCPs in the 4.8.8 tessellation and the 4.6.12 tessellation are NP-complete by reducing from the Tree-Residue Vertex Breaking (TRVB) Problem studied in [5]. Here, *breaking* a degree-n vertex means turning the vertex into n degree-1 vertices that are at the ends of the n edges. The TRVB problem asks that given a planar multigraph M and with some of its vertices marked breakable, is it possible to break some of the breakable vertices so that the resulting graph is a single connected tree. N-Regular Breakable Planar TRVB problem asks that given a planar multigraph with all the vertices degree-n and breakable, is it possible to produce a tree from breaking some vertices. The HCPs in these section reduce from 4-Regular Breakable Planar TRVB problem and 6-Regular Breakable Planar TRVB problem, both of which are NP-complete [5]. The reduction works in this fashion: for any graph M, we will construct a grid graph G of the tessellation so that G has a Hamiltonian cycle if and only if M is breakable.

Theorem 7 *The HCP in grid graphs of the 4.8.8 tessellation is NP-complete.*

Proof. We reduce from the 4-Regular Breakable Planar TRVB problem. When constructing a grid graph G of the 4.8.8 tessellation based on M, we first make a square grid embedding of M, using a method such as the one described in [9]. Then, for each vertex of M, we use the vertex gadget in Figure 9. For the edges in the embedding, we use the edge gadget formed by the boundary vertices of a three-octagon wide strip, as shown in Figure 8. Notice that the edge gadget can shift and turn easily. Due to this flexibility, we can form a graph G based on the embedding using the gadgets.

Now, we will show the constructed graph G has a Hamiltonian cycle if and only if M is breakable. Noticed that if G has a cycle C, both sides of the edge gadgets must be in C and the freedom is only in how to traverse the vertex gadgets. Figure 9 shows two solution to the vertex gadget. The four edge gadgets connect to the vertex on its four sides. Each edge gadgets has two separate paths of vertices that go into the vertex gadget. We call each of these a strand. Note that there are eight single connection edges in the vertex gadgets, each of which is in between a pair of adjacent series. If a cycle exists and a path comes in from a strand, the path must enter one of the two adjacent single edge connections and then connect with the path coming in from another strand. Thus, for a vertex gadget, there are only two kinds of solution: one that has two strands of the same edge connected or one that has two strands of two adjacent edges connected. The first kind is illustrated by the solution on the left, which correspond to a broken vertex in M while the second kind is illustrated by the solution on the right side, which correspond to a unbro-

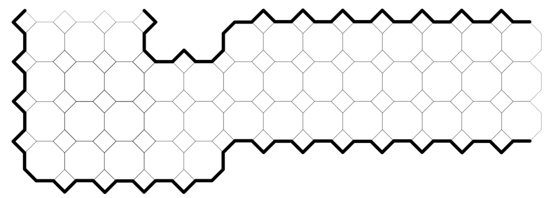

Figure 8: Edge gadget with a turn for 4.8.8. Only edges in bold are present in the gadget.

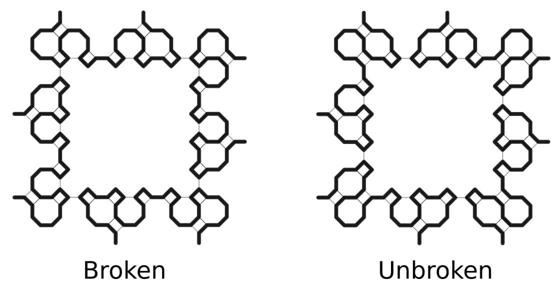

Figure 9: Solutions for the 4.8.8 vertex gadget. The valid paths are shown in bold.

ken vertex in M. To show that G has a Hamiltonian cycle if and only if M is breakable, we apply the reasoning used in the 2017 paper [4]. If M is breakable, then for every broken vertex in M, we traverse through the corresponding vertex gadget using the broken solution; for every unbroken vertex, we traverse through the corresponding vertex gadget using the unbroken solution. Note that after this procedure, the graph produced by breaking M is the same as the region inside the edge gadgets in G. If the graph produced by breaking M is indeed a tree, which is connected and acyclic, then the region inside the edges must also be connected and hole-free, which shows that there is a Hamiltonian cycle. If there is a Hamiltonian cycle in G, the region inside must by connected and hole-free, which then show that the graph M can be broken down to a tree. □

Theorem 8 *The HCP in grid graphs of the 4.6.12 tessellation is NP-complete.*

Proof. We prove that the HCP in 4.6.12 Tessellation is NP-complete by reducing from the 6-Regular Breakable Planar TRVB problem. When constructing a grid graph G in 4.8.8 tessellation based on the multigraph M, we first embed the multigraph in the triangular grid. Then, we use the vertex gadget shown in Figure 11 for every vertex in M and the edge gadget shown in Figure 10 for the edges in M. The edge gadget only includes the boundary vertices of the shape depicted in Figure 10. Because the turning demonstrated in 10 can have turning of 60 and 120 degrees, we can construct the induced subgraph G based on the triangular grid embedding.

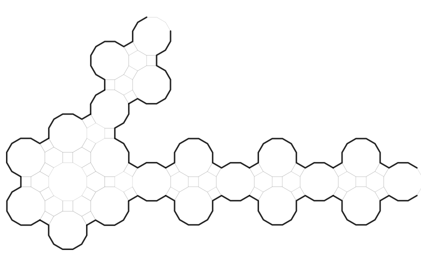

Figure 10: Edge gadget with a turn for 4.6.12

Now, we will show why the constructed graph G has a Hamiltonian cycle if and only if M is breakable. The traversals of the edge gadgets of 4.6.12 tessellation are already set and the only freedom is in how to traverse the vertex gadgets. The six edge gadgets connect to the vertex gadget on the six sides and each edge gadget consists of two strands of vertices. As mentioned in the 4.8.8 tessellation, because of the single edge connections between each pair of adjacent strands, there are only two kinds of traversals for a vertex gadget: the one that has two strands of the same edge connected or the one

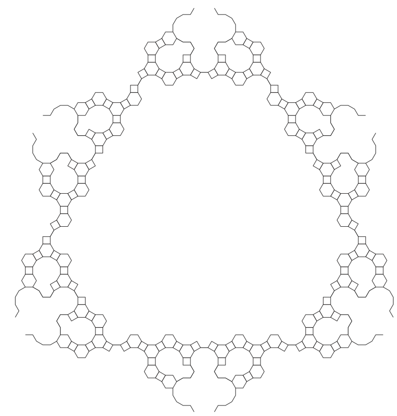

Figure 11: Vertex gadget for 4.6.12

that has two strands of two adjacent edges connected. The first kind is illustrated by the solution in Figure 20, which corresponds to a broken vertex in M. The second kind is illustrated by the solution in Figure 21, which corresponds to an unbroken vertex in M. Just as the argument in 4.8.8 tessellation proof states, the region inside the edge gadgets represents the produced graph after breaking M. □

4 Hamiltonian Cycles with Turns

In this section we explore whether grid graphs contain Hamiltonian cycles which turn at every vertex. In [6], Fekete and Woeginger give a near linear algorithm for finding Hamiltonian paths among a set of points in the plane when the path must turn by 90° at every vertex. We show that this angle-restricted tour problem becomes NP-complete when generalized to 3D, even when we restrict to 3D square grid graphs whose height is only two vertices. We also characterize the complexity of finding always-turning Hamiltonian cycles in triangular grids. See Appendix C and D for these results.

We also investigate a version where each vertex can be visited at most twice, as long as no edges in the cycle overlap. We give linear time algorithms for finding always-turning cycles, as well as double visiting cycles in solid square grid graphs. This question initially came to our attention as special cases of finding Hamiltonian cycles in the 4.8.8 tessellation. There are clear reductions between various problems in this section and restricted versions of that problem in which all vertices around a square pixel are included if any one is included. Although this did not lead to our eventual hardness proof we found the problem to be interesting and well motivated on its own. One can look at this problem as mirroring a problem laid out in a grid where movement is reflected by barriers at 45 degree angles. One can also think of this as counterpart to the discrete milling problem, where our cost function makes turns much less expensive than straight paths.

Theorem 9 *There is a polynomial time algorithm for determining whether a square grid graph admits a Hamiltonian Path which turns at every vertex.*

Proof. See Appendix B for the proof. □

Theorem 10 *The Always-Turning Hamiltonian Path problem in 3D square grid graphs is NP-complete even if the height of the grid is restricted to be* 2 *vertices.*

Proof. The proof is by reduction from from planar max-degree-3 graphs and follows the structure of the original square gird proof. See Appendix C for details. □

4.1 Double Turning in Solid Graphs

Initially inspired by the HCP in the 4.8.8 tessellation grid we consider the following problem. We define the Double Turning Hamiltonian Cycle Problem to be the following: Given a square grid graph, does there exist a cycle in that graph which visits every vertex at least once, never traverses an edge more than once, and turns at every vertex? In particular, this allows the path to visit degree-4 vertices twice, taking a different turn each time. If we consider the square and octagon tessellation in which we have every vertex around a square pixel if any vertex is present around that pixel, then one can see these are equivalent problems.

This section will begin by observing some useful properties of the Hamiltonian path. Then we will connect those to properties of the graph to show that these graphs have a property we call a checkering. Next, we demonstrate that spanning trees of the checkering correspond to Hamiltonian cycles in our graph. Finally, we argue that all of these properties can be checked in polynomial time.

Lemma 11 *If a solid graph admits a double turning Hamiltonian cycle, it also admits such a cycle where all degree-4 vertices are visited twice.*

Proof. If a degree-4 vertex has only one visit then two adjacent edges must be in the path and two adjacent edges must not be in the path. Let us consider some properties of the empty edges. Since each edge must have a partner in the vertex, then each 'path' of empty edges must either connect to degree-3 vertices or be in a cycle. Degree-3 vertices only occur on the boundary since this grid is solid. Thus if we have a path from one degree-3 vertex to another, then the path has gone from one boundary to another and has thus separated two parts of our graph unless those boundary edges were adjacent. This means the empty edges must form a cycle. If this cycle contains any vertices on its interior, then those vertices are disconnected and the cycle cannot be part of a valid solution. Finally a vertex cannot be visited by an empty path more than once, otherwise it is never visited in the actual path. The only cycles in a grid which obey these constraints are single pixels. If there is a pixel without edges, then we pick one vertex arbitrarily to extend the cycle into the pixel. □

With this lemma we can now restrict our examination to the case where all degree-4 vertices are visited twice.

First, we notice that the graph must have even parity on all external boundaries. Given this parity constraint and that the graph is solid, we know that if the graph contains a Hamiltonian cycle then it is composed of some number of full pixels, possibly connected at the corners[2]. We now wish to consider an alternate view of this grid graph. Call the *checkerboard* of this graph the set of alternating pixels in the graph starting with the upper left. We call the other pixels the odd checkering.

Now we will imagine connecting the pixels in the checkerboard and show that the existence of a Hamiltonian cycle depends on its properties. Consider the degree-4 vertices, all of which are visited twice by our prior lemma. There are two configurations of paths, each one connects two diagonally adjacent pixels and separates the other two. We can now think of every degree-4 vertex of our graph as either connecting two adjacent checkered pixels or two adjacent odd checkered pixels. We call this connection a checkering graph.

Lemma 12 *The Double Turning Hamiltonian Cycle Problem in solid grid graphs admits a Hamiltonian cycle if and only if it has a valid checkering and it admits a checkering graph which is a single connected tree.*

Proof. First, we will prove that we can construct a Hamiltonian cycle from a spanning tree of a checkering of the graph. To do so, we will simply visit each of the vertices in an Euler tour order around the spanning tree. Each vertex in the original graph corresponds to a potential edge location in the checkering. We use this term loosely as there may not be vertices in the checkering to connect to. Around each pixel we give the vertices a clockwise ordering. From a vertex we check if that vertex corresponds to an edge in the checkering spanning tree. If not we move clockwise around our current pixel. If it is an edge, we instead consider the pixel we connect to to be our current pixel and move clockwise around that one. We know that every vertex is adjacent to exactly one or two pixels in the checkering and accordingly is visited either once or twice. This process creates a path which never crosses the spanning tree and is free to continue around the entire spanning tree, thus

[2] These corner connections are local cuts and what prevent this graph from being categorized as polygonal.

resulting in a single cycle as desired. An example can be seen in the Appendix.

Now we will argue that if no spanning tree exists then no Hamiltonian cycle exists. If there are two disconnected components of the checkering then this means either there are disconnected pixels in which case either the graph itself is disconnected, the graph is not checkerable, or along all connecting vertices their checkerboard edges were assigned to the odd checkering. The graph must obviously be connected and by the prior parity argument it must be checkerable for it to admit a double turning Hamiltonian cycle. This leaves the case where we have assigned edges in our checkering graph such that it is disconnected. To do so means we would have a path through the odd checkering which separates the two parts of our checkering graph. In the same way that a Hamiltonian path cannot cross edges in the checkerboard graph, it also cannot cross edges in the odd checkering. Thus we have a vertex cut with no paths passing through it, meaning we either have more than one cycle or miss some vertices in our path. □

Now we merely need to show that the checkering and its spanning tree can be found in linear time.

Theorem 13 *The Double Turning Hamiltonian Cycle Problem in solid grid graphs can be solved in linear time.*

Proof. By Lemma 12 we see that deciding if the graph is checkerable and finding a spanning tree of the checkering suffices. See Appendix E for analysis. □

5 Conclusion

In this paper, we have shown that the HCPs in all of the eight semiregular tessellations are NP-complete and shown new upper and lower bounds on finding Hamiltonian paths which always turn in various grids. These generalizations we investigated lead to a large variety of open questions. Most of the restrictions from [8] also apply to the semi-regular tessellation graphs and it would be interesting to know whether solid or super-thin versions of these graphs also admit polynomial time algorithms. We also leave open the questions of the complexity of double turning paths in square grid graphs. In addition, the dual graphs of the tessellation graphs are an obvious next target because of their regular structure and connection to discrete motion planning. One could also look at other general classes of tessellation graphs allowing more general shapes, including higher dimensional structures. We are also rather curious whether anything can be shown about finding Hamiltonian paths in aperiodic tessellation graphs.

There are also other interesting extensions of the always turning paths. The polynomial time proofs only hold for grids in the plane, however the arguments seem like they might lead to algorithms for grids on surfaces of bounded genus. It would be interesting to explore the question on square grids on a torus or other topologically distinct surfaces. In addition, the algorithm for finding double turning Hamiltonian cycles in solid square grids looks related to the number of spanning trees of certain types of graphs, as well as the potential removal of squares of edges. It would be interesting to know if it is computationally tractable to count the number of distinct double turning Hamiltonian cycles and whether it bears nice relation to other combinatorial problems. Finally, this notion of restricted turn paths can be applied to other grids or graphs with appropriate geometry.

Acknowledgments. We want to thank Professor Erik Demaine for useful discussion and feedback on this research.

References

[1] 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. URL: `https://doi.org/10.1137/S0097539703434267`, arXiv: `https://doi.org/10.1137/S0097539703434267`, `doi:10.1137/S0097539703434267`.

[2] Esther M Arkin, Sándor P Fekete, and Joseph SB Mitchell. The lawnmower problem. In *CCCG*, pages 461–466, 1993.

[3] Erik D Demaine, Sarah Eisenstat, and Mikhail Rudoy. Solving the rubik's cube optimally is np-complete. *arXiv preprint arXiv:1706.06708*, 2017.

[4] Erik D. Demaine and Mikhail Rudoy. Hamiltonicity is hard in thin or polygonal grid graphs, but easy in thin polygonal grid graphs. *CoRR*, abs/1706.10046, 2017. URL: `http://arxiv.org/abs/1706.10046`.

[5] Erik D. Demaine and Mikhail Rudoy. Tree-residue vertex-breaking: a new tool for proving hardness. *CoRR*, abs/1706.07900, 2017. URL: `http://arxiv.org/abs/1706.07900`.

[6] Sándor P Fekete and Gerhard J Woeginger. Angle-restricted tours in the plane. *Computational Geometry*, 8(4):195–218, 1997.

[7] Alon Itai, Christos H. Papadimitriou, and Jayme Luiz Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, 1982. URL: `http://dx.doi.org/10.1137/0211056`, `arXiv:http://dx.doi.org/10.1137/0211056`, `doi:10.1137/0211056`.

[8] Esther M. Arkin, Sándor P. Fekete, Kamrul Islam, Henk Meijer, Joseph Mitchell, Yurai Núñez Rodríguez, Valentin Polishchuk, David Rappaport, and Henry Xiao. Not being (super)thin or solid is hard: A study of grid hamiltonicity. 42:582–605, 08 2009.

[9] Markus Schäffter. Drawing graphs on rectangular grids. *Discrete Applied Mathematics*, 63(1):75–89, 1995. URL: `http://www.sciencedirect.com/science/article/pii/0166218X9400020E`, `doi:http://dx.doi.org/10.1016/0166-218X(94)00020-E`.

[10] Giovanni Viglietta. Gaming is a hard job, but someone has to do it! *Theory of Computing Systems*, 54(4):595–621, 2014.

[11] S. Waharte, A. Golynski, and R. Boutaba. On the complexity of routing in wireless multihop network. In *2012 8th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 431–436, Aug 2012. `doi:10.1109/IWCMC.2012.6314243`.

[12] Robert Williams. *The geometrical foundation of natural structure*. Dover New York, 1979.

A Appendix: Semiregular Tessilation NP-Completeness Proofs

Theorem 14 *The HCP in the grid graphs of the 3.3.3.3.6 tessellation is NP-complete.*

Proof. Similar to the 3.4.6.4 tessellation in Section 3, the NP-completeness of the HCP tessellation can also be proven by reducing from the HCP in hexagonal grid. We use the gadgets shown in Figure 12 to simulate the vertices and edges of the hexagonal grid. Now, we can construct a simulated graph G for any hexagonal grid G'. For example, the graph formed by two hexagons can be simulated by the grid in Figure 13.

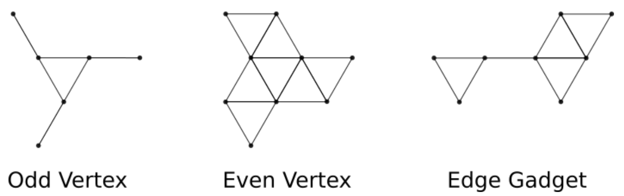

Figure 12: Gadgets for the 3.3.3.3.4 reduction

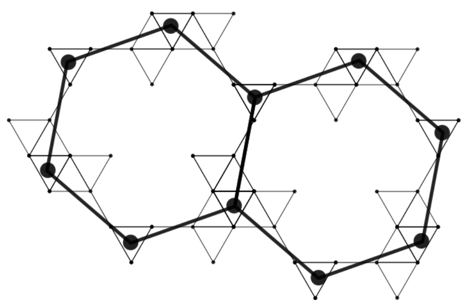

Figure 13: An example of a simulated graph in the 3.3.3.3.4 tessellation

Similar to the gadgets used in Section 3, there are two kinds of traversals for the edge gadget: a cross path that goes from one end to the other end and the return path that begins and finishes on the same end. The following reasoning on why G has a Hamiltonian cycle if and only G' has a Hamiltonian cycle is identical to that of the previous section. If a hexagonal grid G' has a Hamiltonian cycle, we can create a Hamiltonian cycle in G by going through the edge gadgets of the taken edges with cross paths and and the edge gadgets of the untaken edges with return paths. If there is a Hamiltonian cycle in G, each vertex gadget of G must be connected to exactly two cross paths, indicating that there exists a Hamiltonian cycle in G. The reduction is then complete.

□

Theorem 15 *The HCP in the grid graphs of the 3.6.3.6 tessellation is NP-complete.*

Proof. We prove that the HCP in this tessellation is NP-complete by reducing from HCP in hexagonal grid. Using the following vertex gadgets and edge gadget, shown in Figures 14 and Figure 15, for any hexagonal grid G' we can construct a simulated graph G in the tessellation.

Each edge gadget has two kinds of traversals: return paths and cross paths. Return paths begin and end on the same end of the edge while cross paths start and finish on different ends. With some inspection, it is clear that return paths and cross paths are the only two kinds of traversals allowed in the edge gadget. Figure 16 shows a possible return path. Different from those of previous tessellations, the edge gadget here has two kinds of cross paths as shown in Figure 17. Although the two kinds of cross paths start out the same from the odd vertex gadget on the right, they finish in the even vertex on the left differently. The way a cross path connects to an even vertex gadgets dictates which direction it can go next. The upper cross path must turns clockwise when going through the even vertex, allowing it to connect to an upper cross path while the lower one must turn counter-clockwise, allowing it to connect to a lower cross path. By choosing the correct kind of cross paths, any pair of the three edges of the even vertex gadget can be taken by compatible cross paths. By inspection, we can easily see that odd vertex gadget can connect to any pair of the three edges in two cross paths as well.

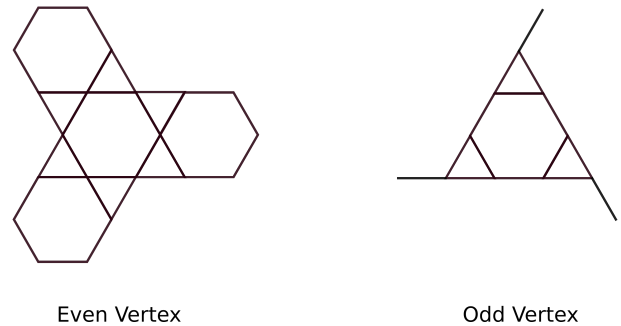

Figure 14: Vertex gadgets for 6.3.6.3

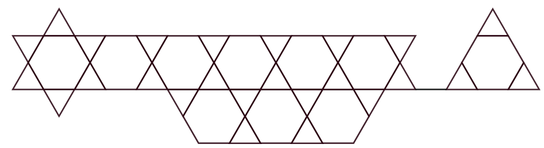

Figure 15: Edge gadgets for 6.3.6.3

Now, we will show that the simulated graph G has a Hamiltonian cycle if and only if the original graph G' has a cycle.

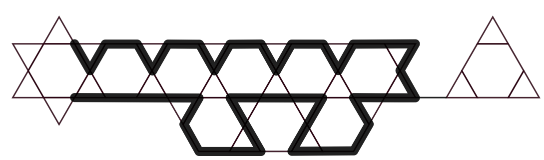

Figure 16: Return path for 6.3.6.3 edge gadget

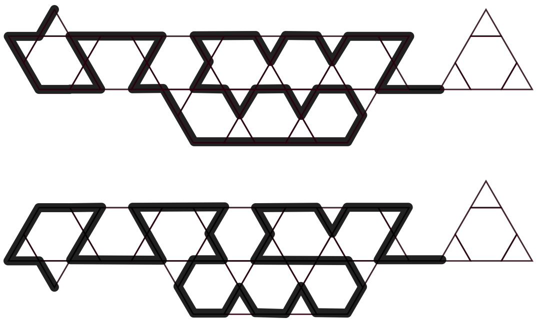

Figure 17: Two Kinds of Cross Paths for 6.3.6.3 edge gadget

If the original hexagonal grid G' has a cycle C', then we go through the edges gadgets representing taken edges in C' with a cross path and those representing untaken edges with a return path. Note that we need to use the correct kind of cross paths so that the choice matches the turning at the vertex. If so, then there is a also a Hamiltonian cycle in G. If the simulated graph G has a Hamiltonian cycle, each vertex gadget must be connected to exactly two cross paths, which indicate that there is a Hamiltonian cycle in G'.

□

Theorem 16 *The HCP in the grid graphs of the 3.12.12 tessellation is NP-complete.*

Proof. This tessellation is composed of dodecagons and triangles. For a hexagonal grid G', we construct a simulated graph G in the tessellation by using the triangles as vertex of G' and the edges in between triangles as the edges of G'. If a Hamiltonian cycle exists in G, each triangle must be connected to two paths that form a 120° angle. Then, there must also be a Hamiltonian cycle in the hexagonal grid G'. If there is a Hamiltonian path in the hexagonal grid G', then there exist one in G.

□

Theorem 17 *The HCP in the grid graphs of the 3.3.3.4.4 tessellation is NP-complete.*

Proof. Similar to the 3.3.4.3.4 tessellation, this tessellation can also be considered as a square grid with extra diagonals. Because its resemblance to square grid, we again use the square grid gadgets. However, if we use the same reduction as in [7], an extra diagonal may disable a pin connection, being an extra edge that connects the odd vertex gadget with the edge gadget. Then, a return path can enter into the odd vertices through this extra edge, causing the former pin connection to no longer function. The connection to the upper edge gadget in an odd vertex gadget shown in Figure 18 is an example of a disabled pin connection. Thus, we will need to modify the reduction.

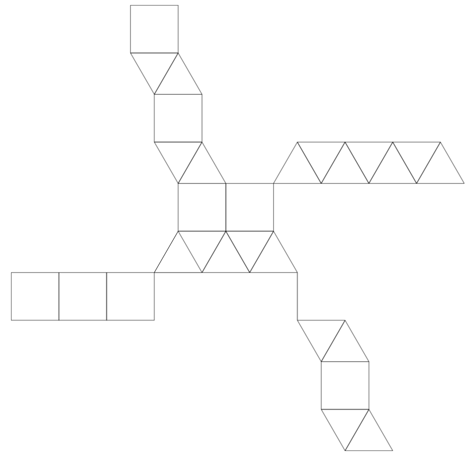

Figure 18: An odd vertex gadget

Although one pin connection may be disabled in a odd vertex gadget, there remains three other functioning pin connections. Because the reduction only requires max degree-3 vertices, there are still ways to make the reduction work. We construct the simulated grid G in the following way. Given a parity preserving square grid embedding of the original max degree-3 bipartite graph G' as mentioned in the 1982 paper [7], we enlarge the embedding by a factor of 3 so that any single segment is at least three segments long and the parities of the vertices are preserved. We then adjust the embedding by replacing every disabled pin connection with a functioning pin connection. Figure 19 shows that if the upper connection is disabled, we use the left or right connection to replace it (the upper row represents embedding before adjustment while the lower row represents embedding after adjustment). Based on the adjusted embedding, we can then construct a simulated graph G using the square grid gadgets. Since the pin connections are all functioning in G, the reduction works.

□

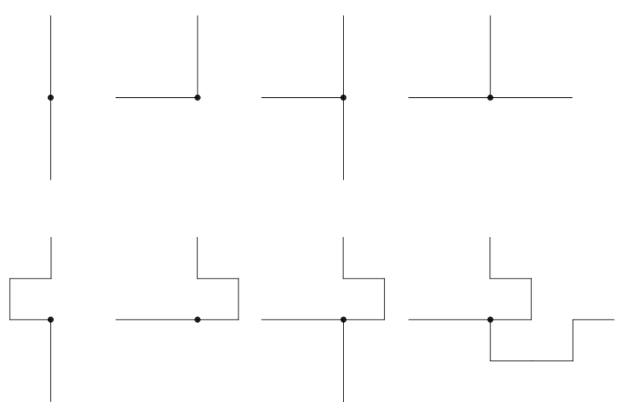

Figure 19: Embedding adjustment

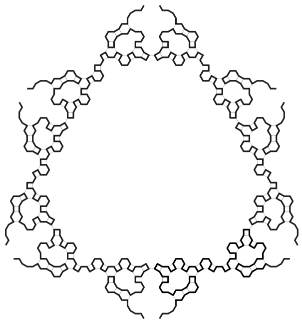

Figure 20: A broken 4.6.12 vertex

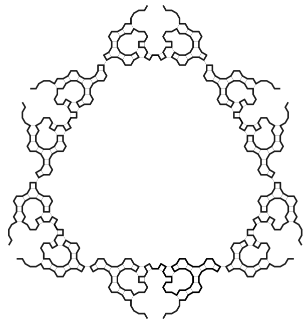

Figure 21: An unbroken 4.6.12 vertex

B Appendix: Turning in Square Grids is Easy

We start with a proof for max-degree-3 square grid graphs because of it's simplicity, and then extend those arguments to handle all square grid graphs.

Theorem 18 *There is a polynomial time algorithm for determining whether a max-degree-3 square grid graph admits a Hamiltonian Path which turns at every vertex.*

Proof. First we notice that any degree-1 vertices, and any degree-2 vertices without a turn make it impossible to have a Hamiltonian cycle. Next, all other degree-2 vertices must have both edges in the cycle. Finally, degree-3 vertices form T intersections. If the path must turn then the middle edge of the T must be included in the cycle.

The only choice remaining is which of the two straight edges in each T intersection is included in the cycle. If one of these two edges lies next to a forced edge, then we know which of the two edges must be included and further that the opposite edge cannot be in the cycle. Now we consider a line of T intersections as in Figure 22. At some point this line must terminate, implying the edge directly opposite the last in the chain does not exist. This implies the last edge must be in the path, which forces the second to last edge to not be in the path and so forth. Thus all T intersections are also forced leading to the following algorithm.

1. If the graph contains any degree-1 vertices, return false.
2. If the graph contains any degree-2 vertices without a turn, return false.
3. Find all degree-2 vertices and mark all edges as being in the path. When you mark an edge as being in a path, follow it to the next vertex and mark the opposite edge as not in the graph. Repeat the process of alternating marking edges as in or not in the cycle until an already marked edge is reached. If the marks disagree, return false, otherwise continue.
4. Find all degree-3 vertices and marks all middle edges as being in the path.When you mark an edge as being in a path, follow it to the next vertex and mark the opposite edge as not in the graph. Repeat the process of alternating marking edges as in or not in the cycle until an already marked edge is reached. If the marks disagree, return false, otherwise continue.

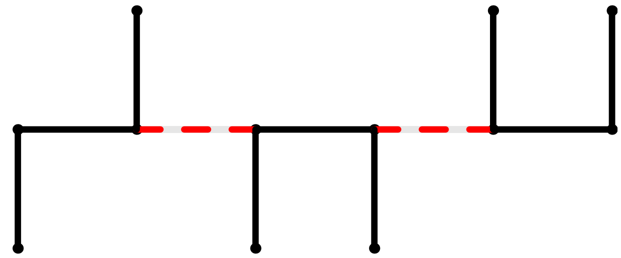

Figure 22: An example row of connected vertices. Forced edges are in bold and forbidden edges are in dashed red.

5. At this point we have marked all edges. Pick a start and check if the edges marked as being in the cycle in fact form a Hamiltonian cycle. If so, return true, otherwise return false.

□

Now we present the main theorem.

Theorem 19 *There is a polynomial time algorithm for determining whether a square grid graph admits a Hamiltonian Path which turns at every vertex.*

Proof. A Hamiltonian Path which turns at every vertex in a grid graph imposes the following constraint: for every vertex except the start and end, precisely one vertical and one horizontal edge must be in the path. This quickly leads to the conclusion that degree-1 and degree-2 vertices either make a Hamiltonian path impossible or forces the edges to be in the path. First, guess the first and last vertex and edge in the path and remove the unused edges next to those vertices from the graph. Now consider a row or column in the graph. In that row we have some number of groups of contiguous vertices which we will consider one at a time. Take the farthest left vertex in this group, it cannot have a left edge. This forces the right edge to be in the Hamiltonian path. Now look at the next vertex. We've already determined it's left edge is in the Hamiltonian path, forcing its right edge to not be in the path. Continuing the next vertex must have its right edge in the path and so on. If there are an even number of vertices, this is consistent, if there are an odd number of vertices then we'll reach a contradiction, declaring a non-existent edge to be in the path. We repeat this process for every group in every horizontal row, and then similarly for every group in every column (starting with the top vertex of each group rather than the left one). We have now assigned every single edge to be either in the path or not. We simply walk the graph to ensure the path is Hamiltonian (aka checking connectivity of our assigned edges) and return the result. □

C Appendix: Turning in 3D Square Grids is Hard

Theorem 20 *The Always-Turning Hamiltonian Path problem in 3D square grid graphs is NP-complete even if the height of the grid is restricted to be 2 vertices.*

Proof. We closely follow the proof that deciding if square grid graphs admit a Hamiltonian path is NP-complete. We reduce from deciding whether planar max-degree-3 graphs admit a Hamiltonian path. We also construct edge gadgets and even and odd vertex gadgets. In this subsection, all figures are two vertices high with paths on the bottom layer represented by black lines and paths on the top layer represented by dotted blue lines.

Edge gadgets are sequences of $2 \times 2 \times 2$ cubes. They admit a forward path, representing an edge taken in the graph, shown in Figure 23. They also admit a return path, shown in Figure 24, representing an edge not taken in the graph. The edges can be turned, as shown in Figures 25 and 26.

Vertex gadgets are represented by $4 \times 8 \times 2$ rectangles. Edges attach across the marked edges e_1 to e_4. Figures 27, 28, and 29 show three different paths through a vertex which will connect any three of the four target edges. Unlike the original grid proof, it is critical that the problem we are reducing from is max-degree-3. Even vertex gadgets attach to the edge gadgets by a $1 \times 2 \times 1$ pair of vertices. If a path enters this pair of vertices, it is then forced to take the edge connecting them. Thus in an even vertex the path can only pass from the vertex to each edge a single time. Since it is max-degree-3, this means precisely two of the adjacent edge gadgets are taken and one is not. Since every odd vertex is connected to an even vertex, this means the odd vertex gadgets must also have precisely two of their adjacent edge gadgets have a taken path. Thus there is only a Hamiltonian cycle if the original graph admits a Hamiltonian cycle.

□

D Appendix: Turning in Triangular Grid Graphs

We now examine the question of Hamiltonian Paths in triangular grid graphs which must turn at every vertex. First, notice that there are now two types of turns: 60° and 120°. It is simple to show the turning Hamiltonian Path problem in triangular grids with 60° is easy and with 120° is NP-complete. For the 120° turns, first notice that we can remove alternating vertices from a triangular grid to leave a hexagonal grid. Further, all remaining edges are 120° turns from each other. Thus we have a very simple reduction from finding Hamiltonian Paths in Hexagonal grid graphs to finding Hamiltonain Paths which always turn 120° in Triangular Grids.

For the case of 60° turns, the answer becomes simple. With out loss of generality, pick two adjacent edges to be in the Hamiltonain Path, such as the two in Figure 30. Now consider the next edge (options show as dotted edges), either it turns to the right and completes the triangle (red edge) or it turns left (blue edge). If it turns left then we once again have two edges of a triangle in our path, only allowing one legal option as seen in Figure 31. Thus, the only Hamiltonian paths which always turn 60 are subsets of a straight zig-zag. Similarly, the only allowed Hamiltonian cycle is a triangle. We've now characterized turning Hamiltonian paths in triangular grid with both polynomial time algorithms and NP-completeness depending on what turns are allowed.

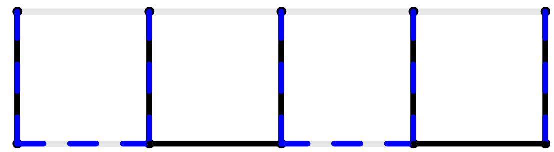

Figure 23: An edge taken in the simulated graph. The path here starts on one side and ends on the other

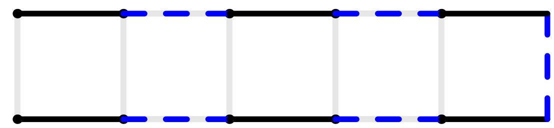

Figure 24: An edge not taken in the simulated graph. The path starts and ends both on the left side.

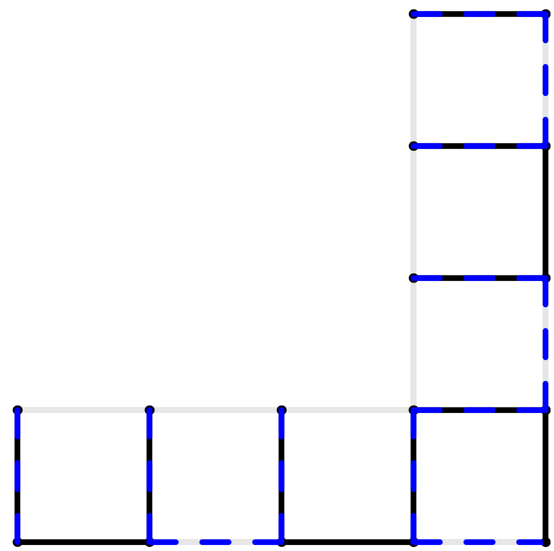

Figure 25: A taken edge being turned.

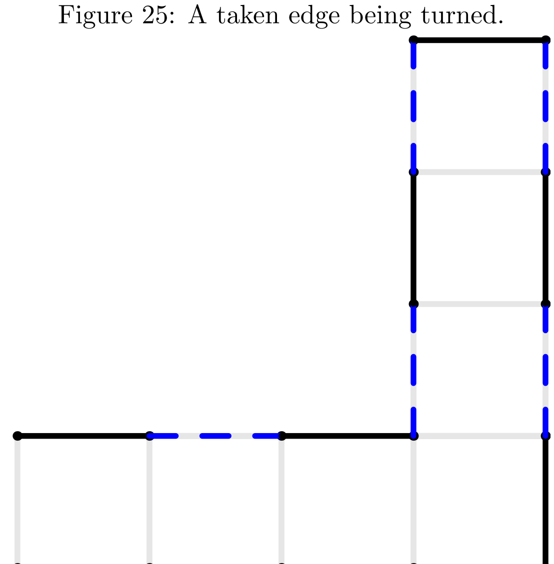

Figure 26: A non-taken edge being turned.

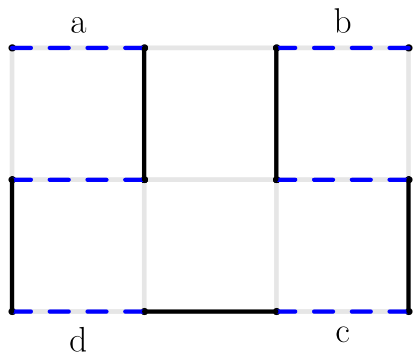

Figure 27: Enters a, crosses c, leaves b.

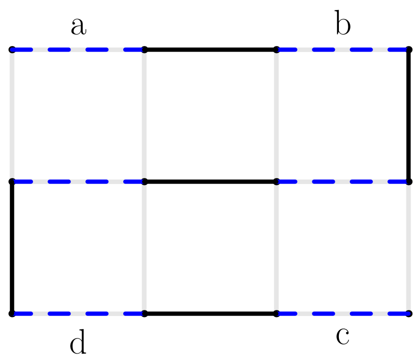

Figure 28: Enters a, crosses b, leaves c

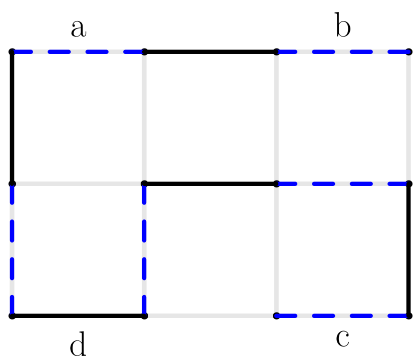

Figure 29: Enters b, crosses a, leaves c.

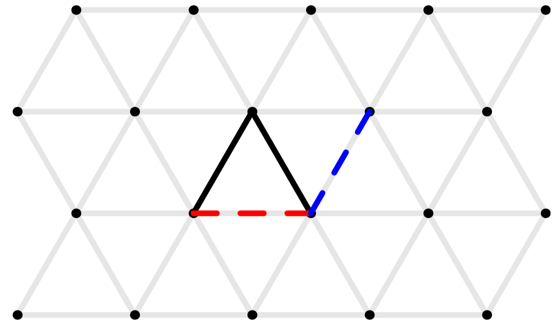

Figure 30: Turns in a triangular grid with forced 60° turns.

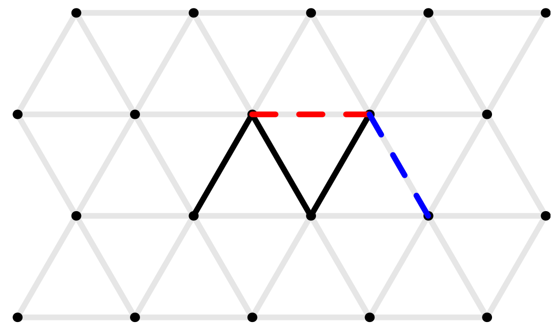

Figure 31: After another step the edges are still forced.

E Appendix: Double Turning in Solid Square Grids

Theorem 21 *The Double Turning Hamiltonian Cycle Problem in solid grid graphs can be solved in linear time.*

Proof. By Lemma 12 we see that deciding if the graph is checkerable and finding a spanning tree of the checkering suffices. We assume we are given the graph embedding. First, we pick the left-most, top-most vertex in the graph and check for the 4-cycle defining the only pixel it is a part of. We construct a vertex in our checkerboard graph for this vertex and associate all four of the vertices around the pixel with it. Now we will perform a breath-first search over the checkered pixels in our graph. From each vertex of our current pixel, look at their exterior edges. If there are two, check whether it has been considered before and if not put that in a queue as a candidate checkerable pixel. If there is only one edge, we maintain a separate list of such degree-3 external edges. First, we check if that edge is in our list, if so we remove the edge and if not we add the edge to the list. Next, pop a candidate pixel off of the queue, verify that all four vertices are there, making it a valid pixel, and recursively check for its neighbors as before. If we ever discover a candidate pixel which is missing a vertex, then one of our properties is violated and we respond that there is no Hamiltonian cycle. Otherwise, we will have constructed a partial checkering and the bfs ordering will give us a spanning tree. Now, we need to take all edges from degree-3 vertices which need to be verified and check that both ends of the edge are vertices which belong to pixels in our checkering. We do the latter simply by checking whether our external degree-3 list is empty because every such edge will be added and then removed the two times it touches a valid pixel. If this is true, return that there exists a Hamiltonian cycle, and if not return false. Verifying each pixel and constructing a new vertex in our checkering takes constant time. Running our bfs touches each pixel, and thus each vertex in our graph a constant number of times. Each extra edge is touched twice. Thus the whole algorithm can be constructed to run in linear time. □

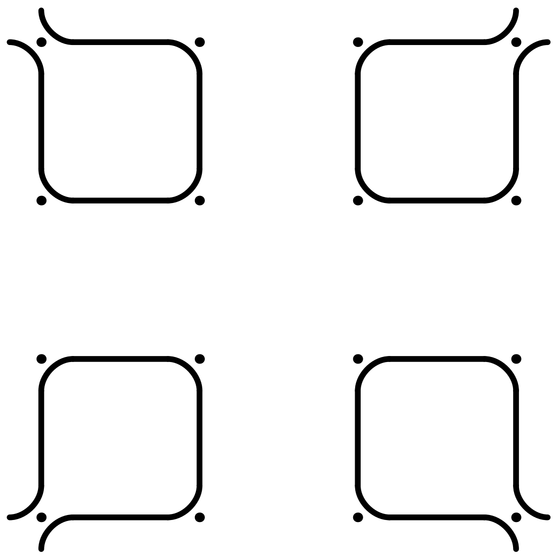

Figure 32: A pixel with all four edges not included in the path.

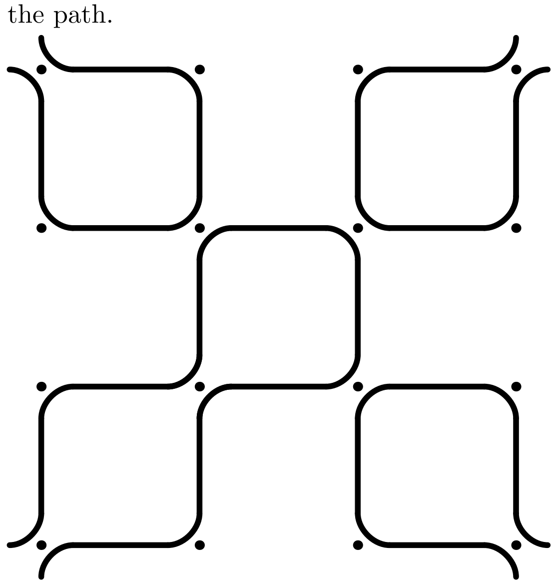

Figure 33: We can augment the path to visit each vertex twice..

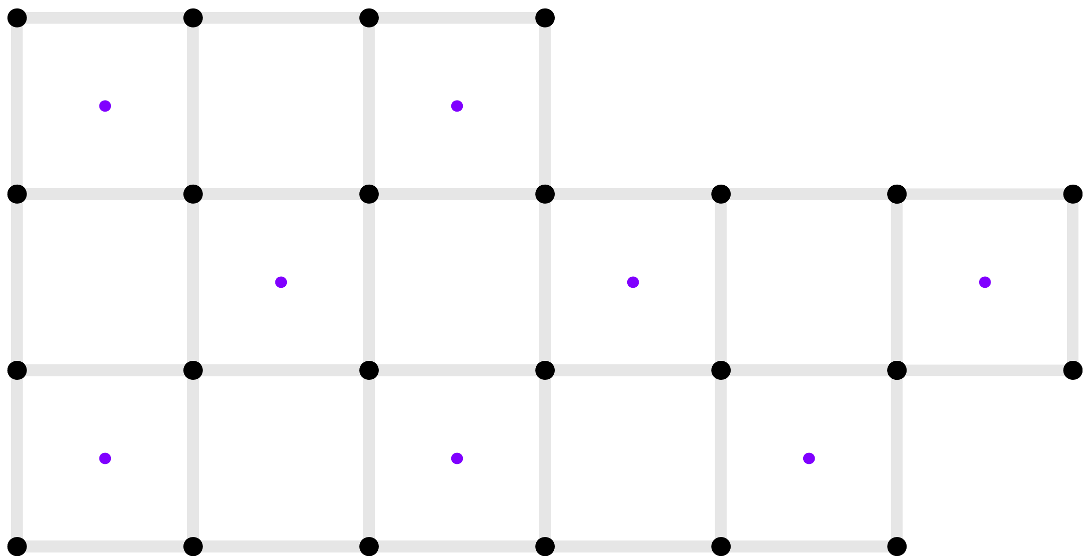

Figure 34: An example solid grid graph with its checkering in purple.

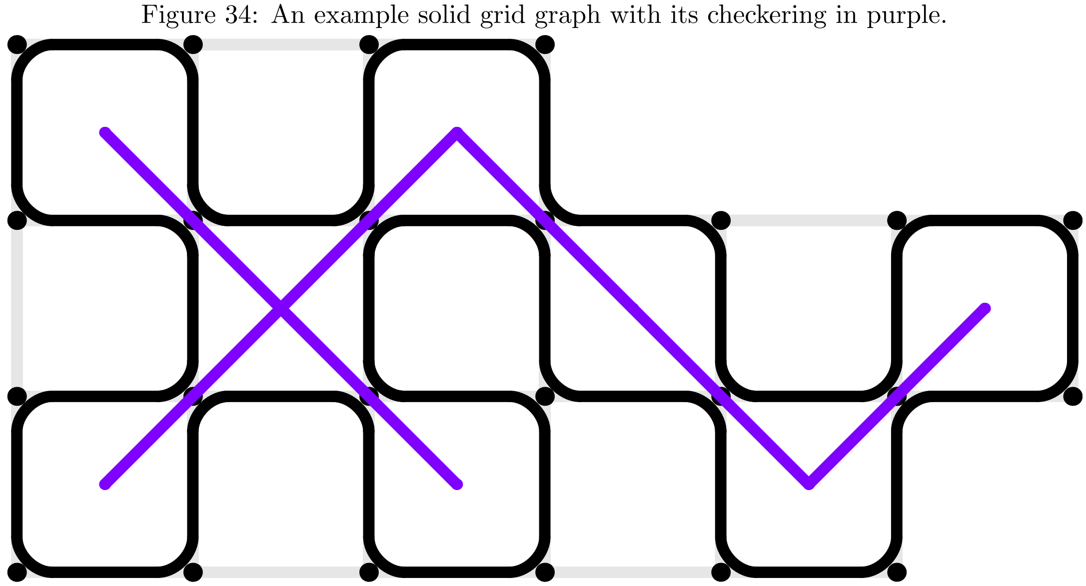

Figure 35: The solution to the example with the spanning tree of its checkering in purple.