Reconfiguration of Hamiltonian Cycles and Paths in Grid Graphs

by

Rahnuma Islam Nishat
B.Sc.Engg., Bangladesh University of Engineering and Technology, 2009
MSc, University of Victoria, 2013

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Computer Science

Reconfiguration of Hamiltonian Cycles and Paths in Grid Graphs

by

Rahnuma Islam Nishat
B.Sc.Engg., Bangladesh University of Engineering and Technology, 2009
MSc, University of Victoria, 2013

Supervisory Committee

Dr. Sue Whitesides, Supervisor
(Department of Computer Science)

Dr. Venkatesh Srinivasan, Departmental Member
(Department of Computer Science)

Dr. Gary MacGillivray, Outside Member
(Department of Mathematics and Statistics)

Supervisory Committee

ABSTRACT

A *grid graph* is a finite embedded subgraph of the infinite integer grid. A *solid grid graph* is a grid graph without holes, i.e., each bounded face of the graph is a unit square. The *reconfiguration problem for Hamiltonian cycle or path in a sold grid graph G* asks the following question: given two Hamiltonian cycles (or paths) of G, can we transform one cycle (or path) to the other using some "operation" such that we get a Hamiltonian cycle (or path) of G in the intermediate steps (i.e., after each application of the operation)? In this thesis, we investigate reconfiguration problems for Hamiltonian cycles and paths in the context of two types of solid graphs: *rectangular grid graphs*, which have a rectangular outer boundary, and *L-shaped grid graphs*, which have a single reflex corner on the outer boundary, under three operations we define, *flip*, *transpose* and *switch*, that are local in the grid.

Reconfiguration of Hamiltonian cycles and paths in embedded grid graphs has potential applications in path planning, robot navigation, minimizing turn costs in milling problems, minimizing angle costs in TSP, additive manufacturing and 3D printing, and in polymer science.

In this thesis, we introduce a complexity measure called *bend complexity* for Hamiltonian paths and cycles in grid graphs, and using those measures we measure complexity of a grid graph G and give upper and lower bounds on the maximum *bend complexity* of an $m \times n$ grid graph. We define three *local* operations, *flip*, *transpose* and *switch*, where *local* means that the operations are applied on vertices that are close in the grid but may not be close on the path or cycle. We show that any Hamiltonian cycle or path can be reconfigured to any other Hamiltonian cycle or path in an $m \times n$ *rectangular* grid graph, where $m \leq 4$, using $O(|G|)$ flips and transposes, regardless of the bend complexities of the two cycles. We give algorithms to reconfigure 1-complex Hamiltonian cycles in a rectangular or L-shaped grid graph G using $O(|G|)$ flips and transposes, where the intermediate steps are also 1-complex Hamiltonian cycles. Finally, we establish the structure of 1-complex Hamiltonian paths between diagonally opposite corners s and t of a rectangular grid graph, and then provide a strategy, based on work in progress, for designing an algorithm to reconfigure between any two 1-complex s, t Hamiltonian paths using *switch* operations.

Contents

List of Figures

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisor Dr. Sue Whitesides for guiding, supervising, and being a very good friend. It has been a wonderful journey thanks to her. Sue provided financial support as well as emotional support to keep me going. She encouraged me to attend conferences to present my results and network with other researchers. She even arranged for my family to be with me at a workshop in Barbados. She was patient with me when I was struggling to get results, nudged me to the right direction when I could not find a way, and encouraged me to push through when I made even a little progress. I am really glad that I decided to have this journey with her.

I would like to thank my parents, Prof M. Muzahidul Islam and Nergis Shahina Akhter, for always believing in me. Special thanks go to my mother for never giving up. She traveled a long way from Bangladesh to Canada multiple times to help me raise my son so that I could concentrate on my studies. Thanks to my other family members as well as my friends Debajyoti Mondal and River Allen for their support and encouragement along the way. During our stay at the Lam Student Family Housing of UVic, we were lucky to have neighbors like Laila, Asad, Fatima, and Rayhan, who made our life enjoyable and hopeful. I am thankful to them for their care and support. Thanks to UVic Child Care for taking good care of my son while I studied.

Finally, I would like to thank the two people who stood by my side till the end, my spouse Sabuj and my son Nahian. Sabuj has been my partner in good times as well as in frustrating times. He has traveled with me to conferences, supported me emotionally when I got frustrated, arranged his work schedules so that I could take time to study. And all I can say about Nahian is that I know no other little boy who understands his mother so well. A part of me has always felt guilty for not spending all my time with him, but another part kept saying that I have to do this so that he can be proud of me.

DEDICATION

Dedicated to all the parents who are striving to be a better role model for their children.

Chapter 1

Introduction

The existence of *Hamiltonian cycles and paths*, named after the Irish mathematician Sir William Rowan Hamilton, is a very well known and well studied topic in mathematics and computer science. Hamilton [59] invented a game he called the *Icosian game* which asks the player to find a *Hamiltonian cycle*, i.e., a cycle that visits every vertex exactly once, in the edge graph of the dodecahedron. This gave rise to the *Hamiltonian cycle or path problem*: given a graph, is there a *cycle*, or a path between two specific vertices of the graph, that visits every vertex of the graph exactly once?

Researchers have explored the existence of Hamiltonian cycles or paths in a graph, and the time and space complexities of algorithms finding such a cycle or path when it exists; they have also enumerated, counted and generated Hamiltonian cycles and paths as we will discuss in Section 1.2. But until recently, the *reconfiguration* of Hamiltonian cycles and paths has remained unexplored.

Reconfiguration problems arise from exploring the *solution space* of some particular problem. For example, a solution space of the Hamiltonian cycle problem for a specific graph G would contain all possible solutions of the problem, i.e., all possible Hamiltonian cycles in G. *Reconfiguration* of the Hamiltonian cycles of G asks the following question: can we transform one Hamiltonian cycle of G to another Hamiltonian cycle of G using some *operations* such that the intermediate steps are also Hamiltonian cycles of G? In this thesis we seek the answer to this question for Hamiltonian cycles and paths in *grid graphs*.

The rest of the chapter is organized as follows. In the next section, we give a formal account of the research questions we solve in this thesis. Section 1.2 discusses background and related work, and Section 1.3 describes some of the many theoretical and practical applications that have motivated our research. Section 1.4 gives the main contributions of this thesis, and presents the organization of the chapters of the thesis for the reader.

1.1 Research Questions

A *grid graph* is a finite subgraph of the infinite integer grid. In this thesis, we assume that the grid graphs are always embedded. When studying the reconfiguration of Hamiltonian cycles and paths in a grid graph, we first try to measure the *complexity* of a grid graph and ask the following questions.

Question 1. How can we measure the *complexity* of a Hamiltonian cycle or path in an embedded grid graph G?

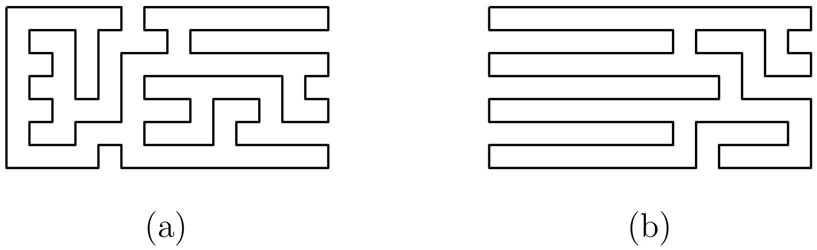


Figure 1.1: How much "more complex" is the Hamiltonian cycle in (a) than the Hamiltonian cycle in (b)?

Question 2. Can we measure the *complexity* of G itself in terms of the *complexities* of the cycles and paths of the graph G?

In Chapter 2 we propose answers to these questions that also give us insight into the *structures* of Hamiltonian paths and cycles in grid graphs. Understanding the structures of Hamiltonian cycles and paths in grid graphs helps us define *operations local in the grid* (see Chapter 2) that we use in the design of algorithms for reconfiguration in later chapters.

Question 3. Can we define *operations* to reconfigure Hamiltonian cycles and paths in grid graphs using properties of grid graphs such as *limited degree* and the fact that the graph is *embedded*? When do the operations preserve *Hamiltonicity*?

Question 4. How is the *complexity* of the cycle or path affected by these *operations*?

Question 5. How are the operations related to each other?

By defining the operations we are going to use to reconfigure one Hamiltonian cycle (or path) of some grid graph G to another, we are actually also implicitly defining the *solution space graph* $\mathcal{G}(G)$, where the vertices of $\mathcal{G}(G)$ are Hamiltonian cycles (paths) of G, and an edge (u, v) between two vertices u and v of $\mathcal{G}(G)$ denotes that one of the *operations we defined* transforms a Hamiltonian cycle (path) represented by u to the other cycle (path) represented by v. We then explore *connectivity* of the solution space graph $\mathcal{G}(G)$ by asking the following questions.

Question 6. Given two Hamiltonian cycles/paths of a grid graph, can we reconfigure one to the other using the operations we defined, i.e., is the solution space graph connected under the operations we defined? How many operations do we need?

Question 7. In a reconfiguration problem for Hamiltonian cycles (paths), if the *complexities* of the starting and ending cycles (paths) are the same, can we give an algorithm that keeps the complexity *unchanged* in the intermediate cycles/paths?

We answer these questions in Chapters 4 and 5 for cycles, and in Chapter 6 we address them for paths. We describe our contributions in more detail in Section 1.4.

1.2 Background

In this section, we discuss previous work on Hamiltonian cycles and paths, and on reconfiguration problems in general.

1.2.1 Knight's Tour and Traveling Salesperson Problems

Historically, the root of Hamiltonian cycle and path problems can be traced back to the *knight's tour* problem: give a sequence of moves for a *knight* on a *chessboard* so that the knight visits every square of the board exactly once. This problem was studied by Euler in 1759 [43] for an 8×8 chessboard. Later researchers studied the problem extensively by considering variations of the size of the board (e.g., 7×7 [40], 4×4 [66], boards with an even number of squares [22], etc.); by considering the problem in which the knight must visit all the squares except a specific square; and by applying various techniques [8, 30, 106, 41, 42]. Takefuji [117] gave a parallel algorithm to solve the knight's tour problem on an $m \times n$ board using neural network computation, while Lin and Wei [89] gave a linear time ($O(mn)$) sequential algorithm to solve the problem for arbitrary m, n.

Another generalization of the Hamiltonian cycle and path problem is the *Traveling Salesperson Problem* or *TSP* for short: given a list of cities and the distance between each pair of cities, what is the shortest route for a traveling salesperson to visit each city and return to to the first city? Although the origin of this problem can be traced back to a German handbook printed in 1832 [86], the problem was not of mathematical interest until Thomas Kirkman and William Hamilton studied it in the later half of the 19th century; a detailed account of their work can be found in the book *Graph Theory 1736–1936* by Biggs, Lloyd and Wilson [10]. In the 1920's and 1930's, Karl Menger popularized the problem. Whitney coined the term "Traveling Salesperson Problem" around 1932 [123]. A chronological account can be found in the article entitled "On the History of Combinatorial Optimization (Till 1960)" by Alexander Schrijver [113]. Since the 1930's, numerous variations of the TSP have been studied by many researchers [34, 93, 4, 85, 90, 47, 63, 58, 45, 32, 11] as a fundamental problem in the theory of computer science. Arora [7] gave a polynomial time approximation scheme for the Euclidean TSP, which had been proved to be an NP-complete problem by Garey et al. [49] and Papadimitriou [105].

1.2.2 Hamiltonian Cycles and Paths in General Graphs

The Hamiltonian cycle problem was one of the first problems to be proved to be NP-complete [48, 38]. In his seminal paper "Reducibility among Combinatorial Problems" in 1972, Richard Karp proved NP-completeness of several problems including the Hamiltonian cycle problem for both directed and undirected graphs [79]. Tait conjectured in 1884 that every three-connected cubic planar graph has a Hamiltonian cycle which was disproved by Tutte in 1946 [118]. Tutte provided a counterexample by constructing a graph of 46 vertices. Later Garey, Johnson and Tarjan [50] showed that the Hamiltonian cycle problem remains NP-complete for planar cubic graphs that are 3-connected, using Tutte's example from [118] as a base for their gadget construction. Plesník [108] proved NP-completeness of the problem for planar digraphs with maximum degree three and degree bound two (i.e., the indegree or outdegree of any vertex is at most two).

Apart from computational complexity results, sufficient conditions were given by Dirac [39], Pósa [109], Bondy [15], Ore [103], Chvátal and Erdős [28, 27], Rahman and Kaykobad [110] among many others. For more detailed surveys please check [56, 9, 82, 21, 33, 87, 65, 13].

1.2.3 Hamiltonian Cycles and Paths in Grid Graphs

Hamiltonian path and cycle problems have been studied in detail in the context of grid graphs as initiated by Luccio and Mugnia in 1978 [91]. In 1982, Itai, Papadimitriou and Szwarcfiter [68] showed that it is NP-complete to decide whether a *general grid graph* (i.e., the boundary of the grid graph is not necessarily a rectangle and the graph may have holes; see Figure 1.2(a)) has a Hamiltonian path or a Hamiltonian cycle. They also gave necessary and sufficient conditions for a rectangular grid graph to have a Hamiltonian cycle, and gave an algorithm to find a Hamiltonian path between any two vertices in a rectangular grid graph if it exists. Recently, Chen et al. [25] improved their algorithm for finding a Hamiltonian path between any two vertices.

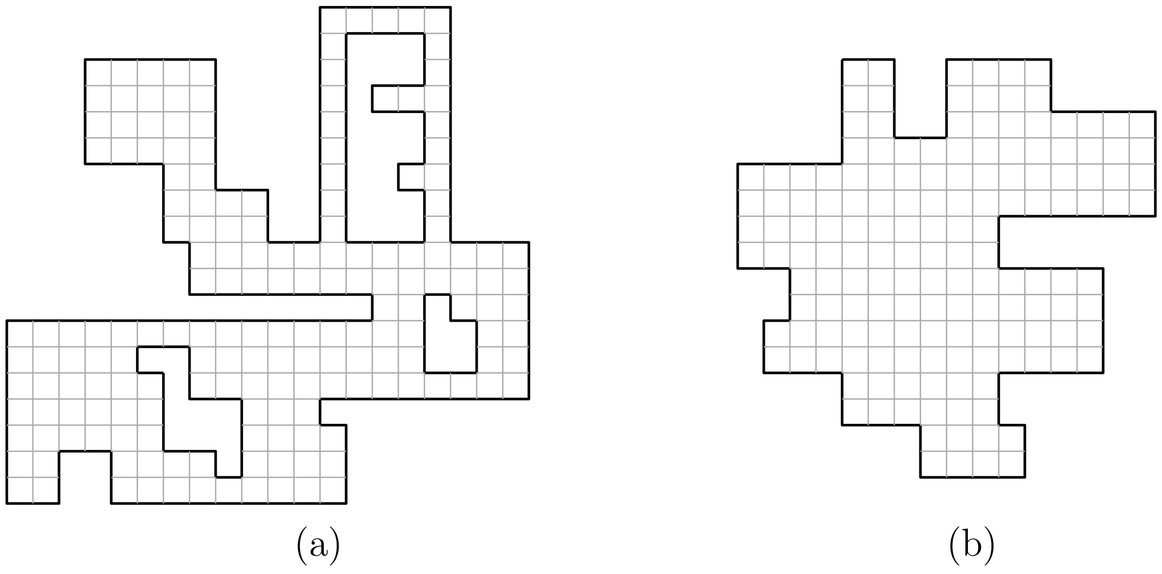


Figure 1.2: (a) A grid graph with holes, (b) a "solid grid graph" with no holes and an arbitrary polygonal boundary.

Itai et al. left the Hamiltonian cycle problem open for solid grid graphs (a grid graph without holes but the outer boundary is not necessarily rectangular; see Figure 1.2(b)). Later Umans and Lenhart [119, 120] gave a polynomial time algorithm to find a Hamiltonian cycle (if it exists) in a solid grid graph using a *cycle merging* technique. They start by finding a 2-*factor* (i.e., a set of disjoint cycles that exactly covers the vertices of the given solid grid graph G). Then they repeatedly transform the 2-factor to reduce the number of components. This process ends either with one component that is a Hamiltonian cycle, or with multiple components, in which case no Hamiltonian cycle exists. Their work is partly based on the findings of Bridgeman in her Bachelor's thesis [20].

Everett [44] solved the Hamiltonian path problem in several specific restrictions of grid graphs by breaking the graphs into rectangular subgraphs. Afrati [1] gave a linear time algorithm for finding Hamiltonian cycles in *restricted grids*, which are left-aligned stacks of decreasing-length grid strips. Cho and Zelikovsky [26] studied *spanning closed trails* containing all the vertices of a grid graph. They showed that the problem of finding such a trail is NP-complete for general grid graphs, but solvable for a subclass of grid graphs that they called *polyminoes*; see Figures 1.3(b)–(c). Sheffield [114] gave an algorithm to determine whether a *polyomino* graph (which is a polymino without holes by Cho and Zelikovsky's definition [26]) with V number of vertices is Hamiltonian or not in $O(V^2)$ steps.

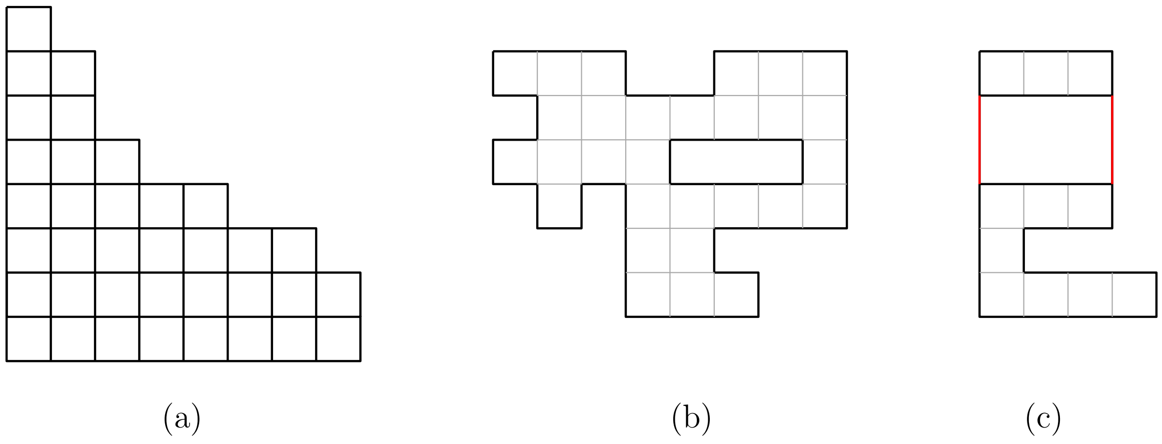


Figure 1.3: (a) Restricted grid graphs studied by Afrati [1]. (b) A *polymino* is a grid graph where every edge of the graph appears on a unit size bounded *cell*, (c) a grid graph that is not a polymino; the red edges do not appear on the boundary of any unit cell of the graph.

Researchers have explored other grids as well. Reay and Zamfirescu [111] and Gordon et al. [55] studied triangular grid graphs that are Hamiltonian. Islam *et al.* [67] showed that the problem of finding a Hamiltonian cycle is NP-complete for hexagonal grid graphs by reduction from the problem of finding a Hamiltonian circuit in a planar bipartite graph of maximum degree 3 (which is also an NP-complete problem). In 2009, Arkin *et al.* [6] studied the Hamiltonicity of grid graphs (i.e. the existence question for a Hamiltonian cycle) and proved several complexity results for different classes of square grids, triangular grids, and hexagonal grids. Zamfirescu and Zamfirescu [125] gave sufficient conditions for a grid graph to be Hamiltonian. Hou and Lynch [64] proved NP-completeness of the Hamiltonian cycle problem for grid graphs of "semiregular tessellations".

Apart from the computational complexity of Hamiltonian cycle problem, combinatorial aspects of the problem have also been studied. Conway *et al.* [31] gave an algorithm to compute self-avoiding walks on $n \times n$ grid graphs for up to 39 steps. Based on Conway's method, Bousquet-Mélou *et al.* [19] gave an algorithm to compute all the self-avoiding walks from the top left corner of a square lattice to the bottom right corner, where $n \leq 19$. Knuth [81] gave an algorithm called SIMPATH to enumerate all simple paths between any two vertices in a graph. Following Knuth's algorithm, Iwashita *et al.* [76] gave an algorithm in 2013 to enumerate all Hamiltonian paths from the top left corner to the bottom right corner of an $n \times n$ grid graph. Their algorithm can compute up to $n = 25$, which is an improvement over previous results. Jacobsen [77] counted the number of *Hamiltonian chains* (defined as sets of k disjoint paths whose union visits each vertex exactly once, where $k = 0$ and $k = 1$ correspond to a Hamiltonian cycle and a Hamiltonian path, respectively) on square lattices in two and three dimensions. In two dimensions, they enumerated chains on an $n \times n$ lattice up to $n = 12$, walks up to $n = 17$, and cycles up to $n = 20$. Pettersson [107] used a dynamic programming technique to enumerate Hamiltonian circuits on a square lattice of size 26.

Niderhausen [99] gave recurrence relations for the number of paths between diagonally opposite corners of a rectangular grid graph that pass through at least I points from a fixed lattice. Myers [97] gave an algorithm to enumerate Hamiltonian cycles in a grid graph of specific size. Göbel [51] gave the exact number of Hamiltonian cycles for a $3 \times n$ grid and the denominator of the generating function for the $4 \times n$ grid.

Kwong and Rogers [84] and Kwong [83] introduced and studied a matrix method to count the number of Hamiltonian cycles in an $m \times n$ grid graph when $m = 4, 5$. Collins and Krompart [29] used their method to give generating functions to count the number of Hamiltonian paths between the lower left corner and upper right corner of an $m \times n$ rectangular grid for $m \leq 5$ and n arbitrarily large. Recently, Keshavarz-Kohjerdi and Bagheri [80] gave necessary and sufficient conditions for the existence of Hamiltonian paths in some "alphabet" grid graphs. Ruskey and Sawada [112] studied "bent Hamiltonian cycles", i.e., Hamiltonian cycles that have each edge in a successive pair of edges in a different dimension, in d-dimensional grid graphs. Stoyan and Strehl [115] gave an algorithm to enumerate the number of Hamiltonian cycles in an $m \times n$ grid graph by systematically constructing a finite automaton.

1.2.4 Reconfiguration Problems

Reconfiguration problems have attracted much attention in recent years. Although arising mostly from theoretical interest, some reconfiguration problems suggest practical applications. For example, the POWER SUPPLY RECONFIGURATION problem [70] deals with a scenario in which a power supply network has to be reconfigured while providing power to all the customers during the transition steps. Insight into the solution space of a reconfiguration problem may be useful in developing heuristics. Other practical applications of reconfiguration problems include maintaining a firewall in a changing network, assigning frequencies in a changing mobile network, and motion planning and robot movement [94, 121].

A very natural reconfiguration problem arises in the popular "sliding block puzzles", where usually a rectangular block has to be slid into a desired position. Hearn and Demaine [61] showed that the problem is PSPACE-hard even when all the blocks (and the free space that allows the sliding) are 1×2 rectangles packed in a rectangle. Bonsma and Cereceda [17] showed that the reconfiguration of k-colorings of graphs, where at each step the color of one vertex is changed to get another valid k-coloring of the graph, is PSPACE-complete for $k \geq 4$, and it remains PSPACE-complete for bipartite graphs and planar graphs. Bonamy et al. [14] studied vertex coloring reconfiguration as well.

A *SAT reconfiguration problem* asks the following question: given two satisfying truth value assignments A and B for a boolean SAT formula $\mathcal{F}$, can we move from A to B by changing the truth value of one variable at a time such that the truth value assignments satisfy $\mathcal{F}$ in the intermediate steps? Gopalan et el. [52] showed that reconfiguration problems of all intractable (i.e., NP-complete) SAT problems are PSPACE-complete. Ito et al. [70] studied complexity of several reconfiguration problems including POWER SUPPLY RECONFIGURATION, INDEPENDENT SET RECONFIGURATION, VERTEX COVER RECONFIGURATION, CLIQUE RECONFIGURATION. In the POWER SUPPLY RECONFIGURATION problem, the initial and final configurations can be represented as subsets G' and G'' of a bipartite graph $G = (U, V, E)$ where one vertex set U represents power stations with fixed capacities and the other vertex set V represents customers with fixed demands, such that the supply from each power station does not exceed its capacity in G' and G''; and at each reconfiguration step one of the customers is moved from one supplier to another such that the move does not violate the capacity restrictions of the power stations. In INDEPENDENT SET RECONFIGURATION, the initial and final configurations are independent sets of a given graph of size at least k (a given integer); and at each step one vertex can be *added* or *removed* such that the size of the independent set is at least $k-1$. The VERTEX COVER RECONFIGURATION and CLIQUE RECONFIGURATION problems have definitions similar to the INDEPENDENT SET RECONFIGURATION problem. Ito et al. [70] showed that although all of the above reconfiguration problems are PSPACE-complete, there are some reconfiguration problems that are in P, such as MINIMUM SPANNING TREE RECONFIGURATION (replace an edge of the given MST with another edge of the graph such that we get another MST) and MATCHING RECONFIGURATION (add or remove one edge from the current matching at a time such that there are at least $k-1$ edges in each intermediate matching).

Reconfiguration of independent sets was studied by Kamiński et al. [78] under three models: *token sliding*, *token jumping* and *token addition/removal*, where *token* means a vertex of the current independent set, and *token sliding* means replacing a vertex with one of its neighbors such that we get another independent set of the same size, and *token jumping* means replacing a vertex with any other vertex of the graph which is not necessarily a neighbor of the vertex removed, and *token addition/removal* means either adding or removing a vertex at any step. Reconfiguration of independent sets was also studied by de Berg et al. [35] where they exchanged exactly k vertices (or tokens) at each step instead of just one, and by Demaine et al. [36]. Haddadan et al. [57] showed that the reconfiguration of dominating set is PSPACE-complete for planar graphs, where each of the initial, the final, and the intermediate dominating sets has cardinality at most k and at each step exactly one vertex is either added or deleted. Mouawad et al. [95] later proved that problem to be W[2]-hard.

Ito et al. [73] studied reconfiguration of cliques and showed that the problem is PSPACE-complete for perfect graphs; Ito et al. [74] studied reconfiguration of vertex covers. Cereceda et al. studied reconfiguration of vertex coloring [23] (where the color of a single vertex is changed at any one step) and also reconfiguration of 3-coloring [24]. Reconfiguration of list *vertex* coloring was studied by Ito et al. [72], Hatanaka et al. [60], and Osawa et al. [104], while Ito and Marcin studied list *edge* coloring reconfiguration [71]. Other reconfiguration problems that have been studied recently include subset sum reconfiguration by Ito and Demaine [69], shortest path reconfiguration by Bonsma [16], Steiner tree reconfiguration by Mizuta et al. [92], reconfiguration of drawings of planar graphs by Alamdaru et al. [3], reconfiguration of maximum-weight matchings by Ito et al. [75], and reconfiguration of dominating sets of cardinality at most k by Mynhardt [98]. For a detailed survey, please see [102].

1.2.5 Reconfiguration of Hamiltonian Cycles and Paths

There have been very few and only recent works on reconfiguration of Hamiltonian cycles. Lignos [88] showed in his PhD thesis that it can be decided in linear time whether a Hamiltonian cycle can be reconfigured to another Hamiltonian cycle using *switch* operations in a graph of maximum degree 5. Takaoka [116] has shown that deciding whether there is a sequence of *switch* operations between two given Hamiltonian cycles is a PSPACE-complete problem for chordal bipartite graphs, strongly chordal split graphs, and bipartite graphs with maximum degree 6. On the other hand, a sequence of switches to reconfigure one cycle to another can be obtained in linear time for bipartite permutation graphs and unit interval graphs. None of the above work considers embedded graphs, however, and hence the structure of the embedded Hamiltonian cycle does not play a role in the reconfiguration.

We introduced the notion of *bend complexity* for Hamiltonian *cycles* in grid graphs, and studied reconfiguration of Hamiltonian cycles of bend complexity 1 in a *rectangular* grid graph [100] and in an *L-shaped* grid graph [101]. We presented our results in international conferences with review committees. This previously published work is described in detail in Chapter 4 and Chapter 5, respectively.

1.3 Motivations and Applications

The idea of applying *local* operations to reconfigure one cycle or path into another was inspired by the study of transforming triangulations by “flipping” one edge at a time. In 1936, Wagner introduced the problem by showing that any “triangulation” (i.e., a planar graph where every face including the outer face is a triangle) can be transformed into a *canonical form*, and hence can be transformed into any other triangulation [122]. Researchers since have studied different aspects of this problem (see [18] for a detailed survey).

Our interest in reconfiguration of Hamiltonian cycles and paths in grid graphs increased as we discovered many theoretical and practical applications as we describe below.

1.3.1 Path Planning and Robot Navigation

Hamiltonian paths and cycles in grid graphs model a variety of applications including robotic path planning problems [54]. Suppose a vacuum cleaning robot is cleaning an indoor environment mapped as a grid graph, and to prevent damage to the floor, the robot must visit each vertex exactly once. The problem then gives rise to a Hamiltonian path or cycle problem [53]. Another application is exploring an area (e.g., a forest, a university campus) with a given map. By overlaying the map with a grid graph, exploring the whole area is reduced to a Hamiltonian cycle problem.

Winter [124] studied modeling routes for robots by considering the cost of turns, for example the turning angle or the necessity to turn. Reconfiguration of Hamiltonian cycles or paths can be applied to find cycles or paths where turn cost is minimized.

1.3.2 Turn Costs and Angle Costs

A fundamental problem of manufacturing is to produce mechanical parts from metal or wooden blocks by clearing areas within a region defined by a specified boundary, called a *pocket*, from the material. The process of clearing is called *milling*. It is a well known computational geometry problem to find an *optimal* tour for the cutting tool or head [62], where the objective function to be optimized may have a variety of definitions.

Arkin et al. considered milling tours where turn cost is minimized [5]. Fellows et al. [46] considered graph milling problems with turn cost minimization. In both these cases, our study in bend complexity and reconfiguration to change bend complexity would be applicable, as it would be in minimizing angle costs in TSP as studied by Aggarwal et al. [2].

1.3.3 Additive Manufacturing

In Additive Manufacturing such as 3D printing, finding a Hamiltonian cycle for the printing head is of utmost importance [96], and the print head moves in a grid. A popular technique is to start with any initial *cycle* and then to reconfigure it to a cycle with desired properties, e.g., least number of turns, least number of switchings between different colored regions, etc. Our reconfiguration technique using operations that are local in the grid may help to keep computation and manufacturing costs relatively low in this scenario.

1.3.4 Application in Chemistry

Enumeration of Hamiltonian paths and cycles has found application in polymer science [37, 12], where Hamiltonian cycles have been used to model what are called "collapsed polymer globules". In this model, the number of Hamiltonian cycles on a lattice (or grid graph) corresponds to the entropy of a polymer system on the grid in what is called "a collapsed but disordered phase".

1.4 Our Contributions

We now list our contributions as well as lay out the organization of the rest of the chapters.

- In Chapter 2, we have the following results.

  1. We study the structure of Hamiltonian cycles in grid graphs. We introduce a measure called *bend complexity* to measure the complexity of a Hamiltonian cycle or path in a rectangular grid graph.
  2. By studying the complexity of Hamiltonian cycles and paths in a grid graph, we determine the *complexity of the embedded graph.* We give upper and lower bounds on the maximum complexity of the graph.
  3. We define three operations, *flip*, *transpose* and *switch*, that are *local in the graph* for reconfiguration. Note that Lignos [88] and Takaoka [116] did not take locality in the graph into consideration.

- In Chapter 3, we show that the solution space of the Hamiltonian cycle and path problems in *rectangular* grid graphs is connected under our *local operations* when one of the grid dimensions is less than 5.

- In Chapter 4, we show that any Hamiltonian cycle with *bend complexity* 1 can be reconfigured to any other such cycle using a linear number of flips and transposes in a rectangular grid graph, while preserving the bend complexity in the intermediate steps.

- In Chapter 5, we achieve similar results for L-shaped grid graphs, i.e. grid graphs without holes and an L-shaped polygon tracing the outer boundary.
- In Chapter 6, we establish the structure of a Hamiltonian path P of bend complexity 1 from the top left corner to the bottom right corner of a rectangular grid graph by providing structure theorems (Theorems 6.1, 6.2 and 6.3). We then give a research program to design an algorithm to reconfigure any such path to any other such path using switch operations.
- Chapter 7 summarizes our results, discusses some work in progress, and provides some interesting open problems arising from our work.

Chapter 2

Grid Graphs: Definitions

A *grid graph* is a finite embedded subgraph of the infinite two-dimensional integer grid. A *solid grid graph* is a grid graph without holes, i.e., each bounded face of the graph is a unit square. The *boundary* of a solid grid graph is the boundary cycle of the outer face of the grid graph. In this thesis, we discuss solid grid graphs with two specific boundary shapes: *rectangular* and *L-shaped*.

An $m \times n$ *rectangular grid graph* (or $m \times n$ *grid graph* for short) is a solid grid graph such that the outer boundary is an $(m-1) \times (n-1)$ rectangle as shown in Figure 2.1(a). An *L-shaped grid graph* is a solid grid graph whose boundary is L-shaped with a single *reflex corner* as shown in Figure 2.1(b).

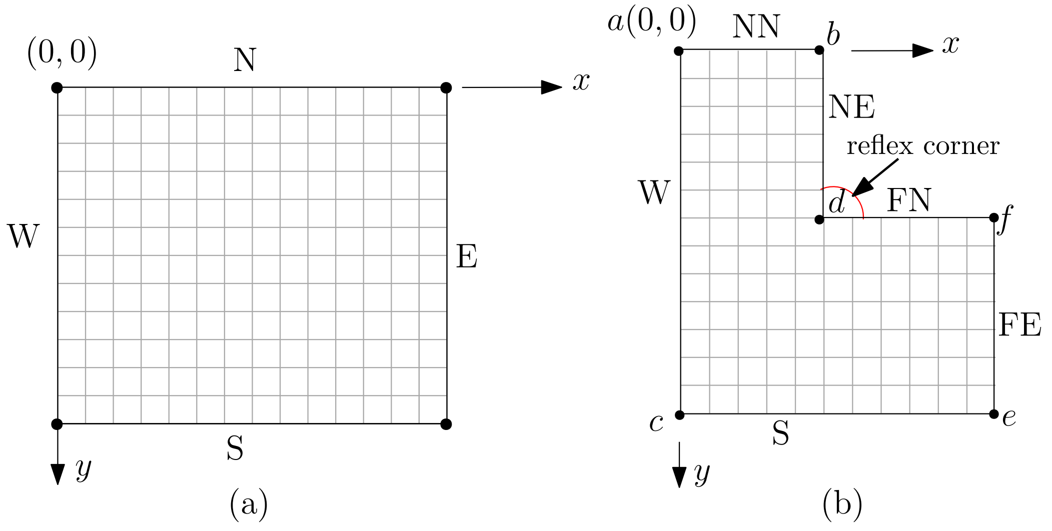


Figure 2.1: (a) A rectangular grid graph embedded on the grid. (b) An L-shaped grid graph with labeled corners and boundary edges.

In this chapter we do the following.

1. In Sections 2.1 and 2.2, we define some terminology for rectangular and L-shaped grid graphs, and for Hamiltonian cycles and paths, respectively.

2. In Section 2.3, we define three operations *flip*, *transpose* and *switch* that are "local" in the grid. These operations are used in reconfiguration algorithms in later chapters (namely Chapters 3, 4, 5, and 6).

3. In Section 2.4, we give a new complexity measure for Hamiltonian cycles and paths in grid graphs that we call *bend complexity.* Using this complexity measure we define *bend complexity of a grid graph.* We then give upper and lower bounds on the "bend complexity of a rectangular grid graph".

2.1 Grid Graph Terminology

Let G be a grid graph, either rectangular or L-shaped. Throughout this thesis, we assume that G is embedded on the integer grid such that the top left corner vertex is at $(0, 0)$. We also assume that the positive y direction is downward and that the positive x direction is to the right, as shown in Figure 2.1. A vertex of G with integer coordinates (x, y) is denoted by $v_{x,y}$. For a vertex v of G, $x(v)$ and $y(v)$ denote its x– and y–coordinates, respectively.

2.1.1 Subgrids

Let G' be an embedded subgraph of G such that G' is either a rectangular or an L-shaped solid grid graph. We call G' a *subgrid* of G. If G' has a rectangular boundary, we call it a *rectangular subgrid* of G; otherwise, G' has an L-shaped boundary and we call it an *L-shaped subgrid* of G. See Figure 2.2.

A *cell* is the 1×1 square outlined by a bounded face of G.

2.1.2 Tracks

Row j is the subgraph of G induced by the set of vertices of G with y-coordinate j, and *Column* i is the subgraph of G induced by the set of vertices of G with x-coordinate i. A *horizontal track* t^h_j is a rectangle determined by the pair of Rows j and $j+1$ (including the boundary vertices and edges) that is inside G; a *vertical track*

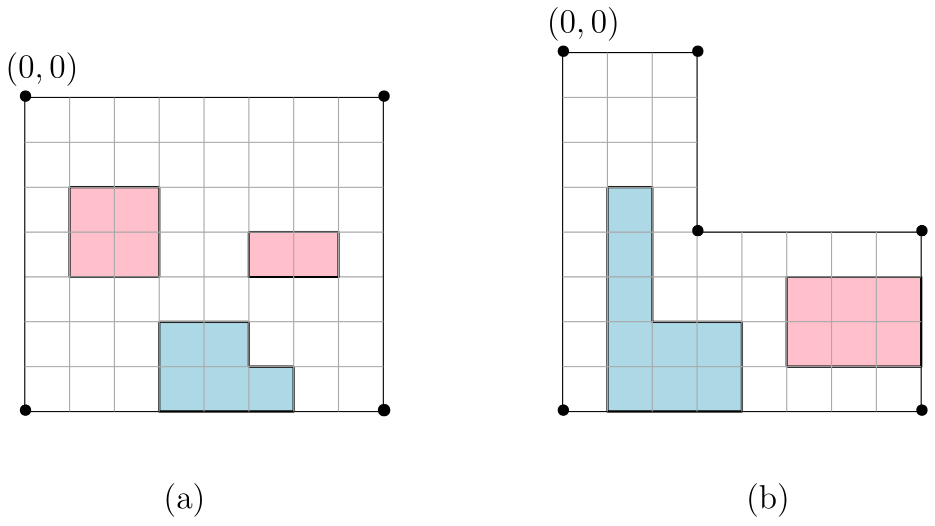


Figure 2.2: Examples of subgrids: (a) a rectangular grid graph, and (b) an L-shaped grid graph. L-shaped subgrids are shown in blue and rectangular subgrids are shown in pink.

t_i^v is the rectangle determined by the pair of Columns i and $i+1$ that is inside G. See Figure 2.3.

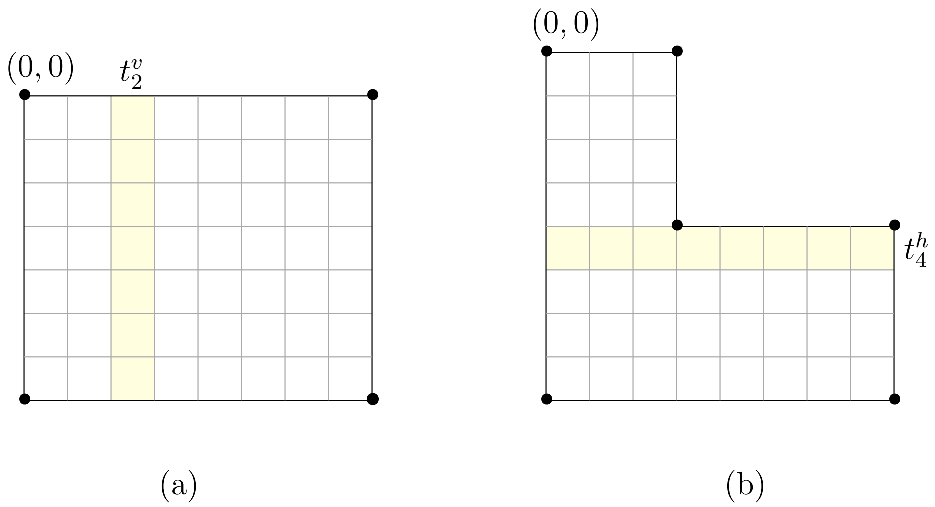


Figure 2.3: (a) A vertical track t_2^v in a rectangular grid graph. (b) A horizontal track t_4^h in an L-shaped grid graph.

The *interior of a track* is the interior of the rectangle that defines the track.

2.1.3 Boundaries of the Grid Graph

The *boundary* of G is the boundary cycle of the outer face of G. Note that the boundary of G is the only face boundary of G that may have more than four vertices. If G is a rectangular grid graph, then we call Columns 0 and $n-1$ the west (W) and east (E) boundaries, and Rows 0 and $m-1$ the north (N) and south (S) boundaries. See Figure 2.1(a).

On the other hand, if G is an L-shaped grid graph, then we call Columns 0 and $x(e)$ the *west (W)* and *far-east (FE) boundaries*; Rows 0 and $y(e)$ are the *near-north (NN)* and *south (S) boundaries*. The vertices on Column $x(d)$ from b to d (inclusive) comprise the *near-east (NE) boundary* and the vertices on Row $y(d)$ from d to f (inclusive) comprise the *far-north (FN) boundary.* See Figure 2.1(b).

2.2 Hamiltonian Paths and Cycles

In this thesis we give algorithms to reconfigure Hamiltonian cycles in rectangular and L-shaped grid graphs (Chapters 4 and 5, respectively). Using the reconfiguration results of those two kinds of cycles we design algorithms to reconfigure Hamiltonian paths between the top-left and bottom-right corners s and t of a rectangular grid graph (Chapter 6). We now define the terminology we use throughout the thesis.

Definition 2.1 (Hamiltonian cycle)**.** *A* Hamiltonian cycle *in a grid graph is a cycle that visits each vertex of the graph exactly once.*

Definition 2.2 (s, t Hamiltonian path)**.** *An* s, t Hamiltonian path *in an* $m \times n$ *rectangular grid graph* G *is a path from the top-left corner vertex* $s = v_{0,0}$ *to the bottom-right corner vertex* $t = v_{n-1,m-1}$ *of* G *that visits each vertex of* G *exactly once.*

In this thesis, any Hamiltonian path we discuss has endpoints at s and t.

2.3 Local Operations

In this section, we define three operations on Hamiltonian cycles and paths that are *local* on the grid. By *local* we mean that the operations are performed on vertices and edges that are close in G but may be far apart on the Hamiltonian cycle/path. Let G be an $m \times n$ grid graph and let C be a Hamiltonian cycle of G. We define the first two of our three operations as operations on cycle C, noting that these operations are also applicable to paths of G.

2.3.1 Flip

We define some terminology needed to define the *flip* operation.

Definition 2.3 (Flippable subgrid)**.** *Let G' be a 3×2 or a 2×3 rectangular subgrid of G, where the vertices a, b, d, f, e, c appear on the boundary of the outer face of G' in clockwise order, and the corner vertices of G' are a, b, f, e. Let C be a Hamiltonian cycle of G such that only the edges $(a, c), (c, d), (d, b)$ and (e, f) of G' are included in C. We call G' a* flippable subgrid *of G with respect to C.*

Definition 2.4 (Flip)**.** *Let G' be a flippable subgrid of G in C. A* flip *operation on C in G' is performed by replacing the path a, c, d, b with edge (a, b), and by replacing edge (e, f) with path e, c, d, f.*

We denote a flip operation on C in the subgrid G' by $Flip(a, f)$ or $Flip(b, c)$, where $\{a, f\}$ and $\{b, c\}$ are the two pairs of diagonally opposite corner vertices of G'. See Figure 2.4(a). We now prove some properties of flip.

Lemma 2.1. *Let G' be a flippable subgrid of G with respect to C. Then a flip operation on C in G' gives another Hamiltonian cycle C' of G.*

Proof. Without loss of generality, assume that G' is the 3×2 subgrid shown in Figure 2.4(a). A flip on G' is performed in two steps. First, replacing the path a, c, d, b with edge (a, b) gives a Hamiltonian cycle on $G - \{c, d\}$. The next step is replacing edge (e, f) with path e, c, d, f, which gives a Hamiltonian cycle C' of G. See Figure 2.4(b). □

Observation 2.1. *Let C be a Hamiltonian cycle of G and let G' be a flippable subgrid of G with respect to C. Let C' be a Hamiltonian cycle of G obtained by applying a flip on C in G'. Then G' is flippable with respect to C' and applying a flip on C' in G' gives back the Hamiltonian cycle C of G.*

2.3.2 Transpose

We define some terminology needed to define the *transpose* operation.

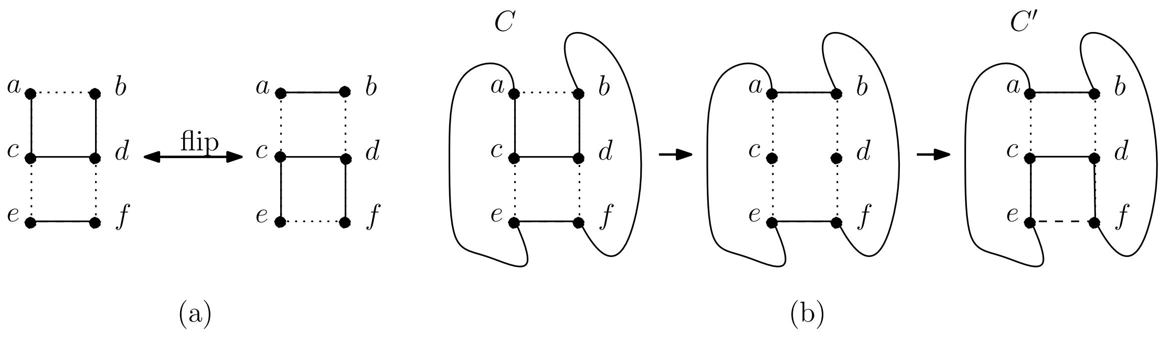


Figure 2.4: (a) A flip on a Hamiltonian cycle C in a grid graph. (b) Illustration for the proof of Lemma 2.1.

Definition 2.5 (Transposable subgrid)**.** *Let G'' be a 3×3 rectangular subgrid of G, where the vertices a, b, g, h, i, f, e, c appear on the boundary of the outer face of G'' in clockwise order, and the corner vertices of G'' are a, g, i, e. The vertex d of G'' is the only vertex that does not appear on the boundary of the outer face of G''. Let C be a Hamiltonian cycle on G such that the edges $(a, c), (f, d), (d, b), (b, g), (g, h), (h, i)$ are included in C. Either or both or neither of the edges (e, c) and (e, f) might be on C. We call G' a* transposable subgrid *of G with respect to C with* pin vertex *e.*

Now we define our second operation, *transpose*. The terminology is roughly inspired by matrix terminology as we interchange the roles of rows and columns. Here, we are rotating around the diagonal (c, g), without affecting the grid edges incident to the *pin vertex e*. The pin vertex pinpoints those grid edges in the transposable subgrid G' that are not affected by the transpose operation.

Definition 2.6 (Transpose)**.** *Let G'' be a transposable subgrid of G with respect to C with pin vertex e. A* transpose *operation on C in G'' is performed by replacing the path f, d, b, g, h, i with edge (f, i), and by replacing the edge (a, c) with path a, b, g, h, d, c.*

We denote a transpose operation on C in the transposable subgrid G'' with the pin vertex e by $Xpose(e, g)$, where g is the corner of G' that is diagonally opposite of e. See Figure 2.5(b). We now prove some properties of transpose.

Lemma 2.2. *Let G'' be a transposable subgrid of G with respect to C. Then a transpose operation on C in G'' gives another Hamiltonian cycle C' of G.*

Proof. Without loss of generality, assume that G'' is the 3×3 subgrid shown in Figure 2.5(a) with pin vertex e, and the vertices of G appear on C in the cyclic order

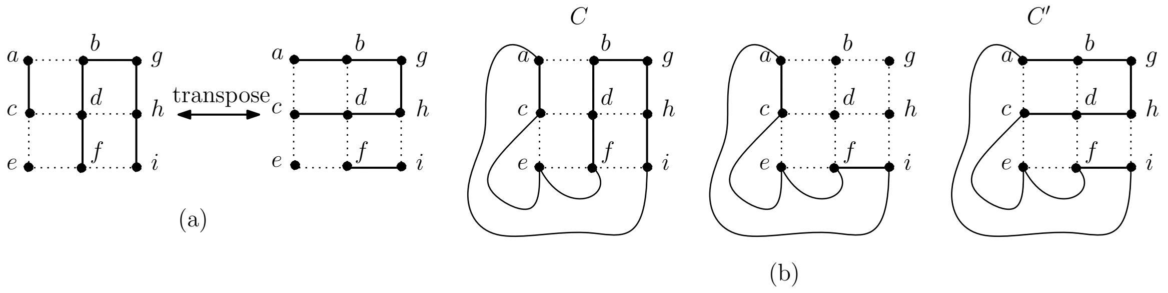


Figure 2.5: (a) A transpose on a Hamiltonian cycle C in a grid graph. (b) Illustration for the proof of Lemma 2.2.

shown in Figure 2.5(b). A transpose on G'' is performed in two steps. First, replacing the path f, d, b, g, h, i with edge (f, i) gives a Hamiltonian cycle on $G - \{b, g, h, i\}$. The next step is replacing edge (a, c) with path a, b, g, h, d, c, which gives a Hamiltonian cycle C' of G. See Figure 2.5(b). □

Observation 2.2. *Let C be a Hamiltonian cycle of G and let G'' be a transposable subgrid in C with pin vertex e. Let C' be a Hamiltonian cycle of G obtained by applying a transpose on C in G'. Then G'' is transposable with respect to C' with pin vertex e, and applying a transpose on C' in G'' with the same pin vertex e gives back the Hamiltonian cycle C of G.*

2.3.3 Switch

Let G be a rectangular grid graph. We first define some terminology needed to define our third local operation, *switch*.

Definition 2.7 (Cycle-Path Cover)**.** *A* cycle-path cover $\mathbb{P}$ *of G is a set of cycles and paths that collectively covers all the vertices of G.*

Definition 2.8 (Switchable cell)**.** *Let $\mathbb{P}$ be a cycle-path cover of G. Let f be a cell of G such that $\mathbb{P}$ covers exactly two edges of f, and these are not incident to a common vertex. Then f is a* switchable cell.

We now make a definition of *switch* based on the definition of *switch* given by Lignos [88] and Takaoka [116], but in contrast to their definition, our switch is applied locally to a cell of an *embedded* grid graph.

Definition 2.9 (Switch). *Let $\mathbb{P}$ be a cycle-path cover of G. Let f be a switchable cell of G with the vertices a, b, d, c on the boundary of f in clockwise order, and let the edges (a, c) and (b, d) be the only edges of f that are in $\mathbb{P}$. Then a* switch operation on $\mathbb{P}$ in f *replaces (a, c) and (b, d) with (a, b) and (c, d).*

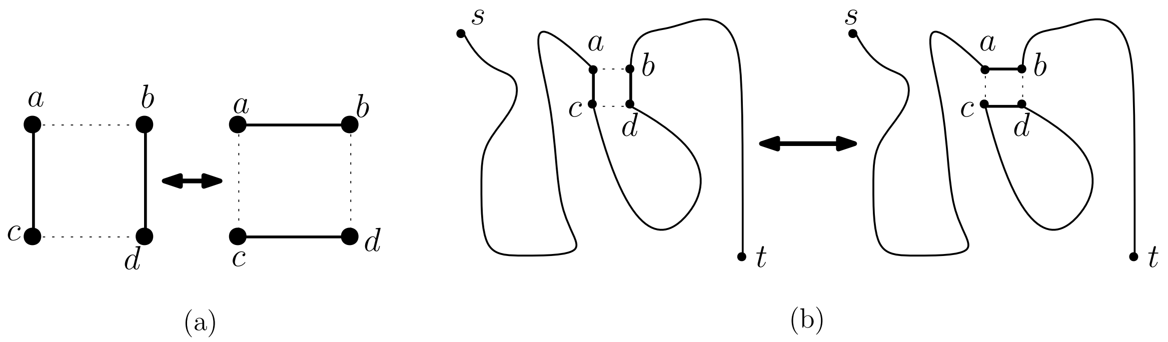


Figure 2.6: (a) A *switch* replaces edges (a, c) and (b, d) with (a, b) and (c, d). (b) A switch on an s, t Hamiltonian path $P = \mathbb{P}$ gives a cycle-path cover $\mathbb{P}'$ consisting of one path from s to t and one cycle.

See Figure 2.6 for an example of switch. We now observe some properties of switch.

Lemma 2.3. *Let P be an s, t Hamiltonian path of G and let f be a switchable cell of G. Then a switch operation in f gives a cycle path cover of G consisting of a single cycle and a single path from s to t.*

Proof. Without loss of generality, assume that f is as shown in Figure 2.6(a). We consider two cases.

Case 1. If neither of s and t is incident to f, we assume without loss of generality that $P = s, \dots, a, c, \dots, d, b, \dots, t$. Removing the two edges (a, c) and (b, d) gives us a cycle-path cover of G consisting of three paths $s, \dots, a$; $c, \dots, d$; and $b, \dots, t$. Adding the edge (a, b) gives a cycle path cover of G consisting of two paths $s, \dots, a, b, \dots, t$; and $c, \dots, d$. Then adding the edge (c, d) connects the two endpoints of the second path to make it a cycle. Therefore, the final cycle-path cover of G consists of a single path from s to t and a single cycle.

Case 2. Suppose that $s = a$. The case when $t = d$ is similar. Without loss of generality assume that $P = s, c \dots, d, b, \dots, t$. Removing the two edges $(s = a, c)$ and (b, d) gives us a cycle-path cover of $G - s$ consisting of two paths $c, \dots, d$, and $b, \dots, t$.

Adding the edge (s, b) gives a cycle path cover of G consisting of two paths $s, b, \ldots, t$; and $c, \ldots, d$. Then adding the edge (c, d) connects the two endpoints of the second path to make it a cycle. Therefore, the final cycle-path cover of G consists of a single path from s to t and a single cycle. □

Lemma 2.4. *Let $\mathbb{P}$ be a cycle path cover of G consisting of a single cycle C and a single path P from s to t. Let f be a switchable cell of G such that each of C and P covers one edge of f. Then a switch operation in f gives an s, t Hamiltonian path of G.*

Proof. Without loss of generality, assume that the vertices a, b, d, c appear on the boundary of f in clockwise order, and let C and P cover the edges (c, d) and (a, b), respectively. See Figure 2.6(b). We consider two cases.

Case 1. If neither of s and t is incident to f, we assume without loss of generality that $P = s, \ldots, a, \ldots, b, \ldots, t$. Removing the two edges (a, b) and (c, d) gives us a cycle-path cover of G consisting of three paths $s, \ldots, a$; $b, \ldots, t$; and $c, \ldots, d$. Adding the edge (a, c) gives a cycle path cover of G consisting of two paths $s, \ldots, a, c, \ldots, d$; and $b, \ldots, t$. Then adding the edge (b, d) connects the two paths giving the s, t Hamiltonian path $s, \ldots, a, c, \ldots, d, b, \ldots, t$ of G.

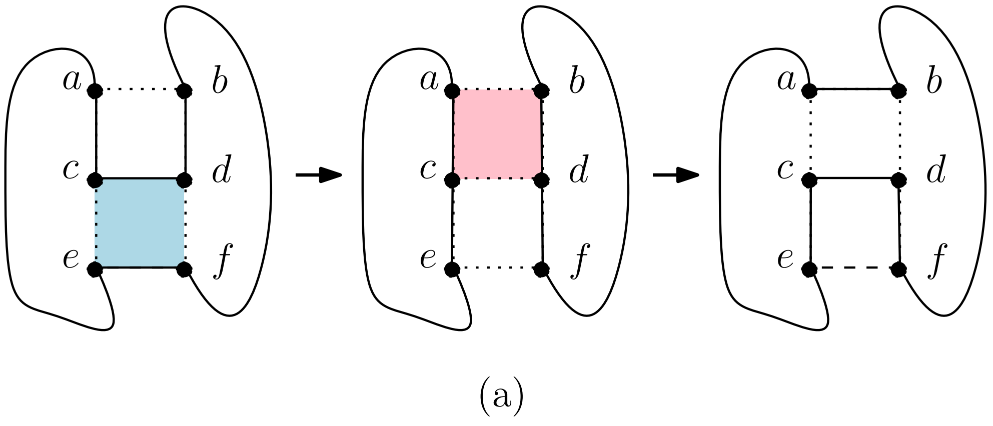
Case 2. Suppose that $s = a$. The case when $t = d$ is similar. Without loss of generality assume that $P = s, b, \ldots, t$. Removing the two edges $(s = a, b)$ and (c, d) gives us a cycle-path cover of $G - s$ consisting of two paths $c, \ldots, d$; and $b, \ldots, t$. Adding the edge (s, c) gives a cycle path cover of G consisting of two paths $s, c, \ldots, d$ and $b, \ldots, t$. Then adding the edge (b, d) connects the two paths to give the s, t Hamiltonian path $s, c, \ldots, d, b, \ldots, t$ of G. □

2.3.4 Relationships among Local Operations

We now show that the operations flip and transpose can be described as a pair switch operations. Although we describe the operations in the context of Hamiltonian cycles, they are applicable in the context of Hamiltonian paths as well.

Relationship between Flip and Switch

Let C be a Hamiltonian cycle of G with the 3×2 flippable subgrid G' as shown in Figure 2.7(a). The only switchable cell in G' is shown in blue. Then we can perform a *flip* in G' by applying two switches. The first switch is applied in the blue cell which



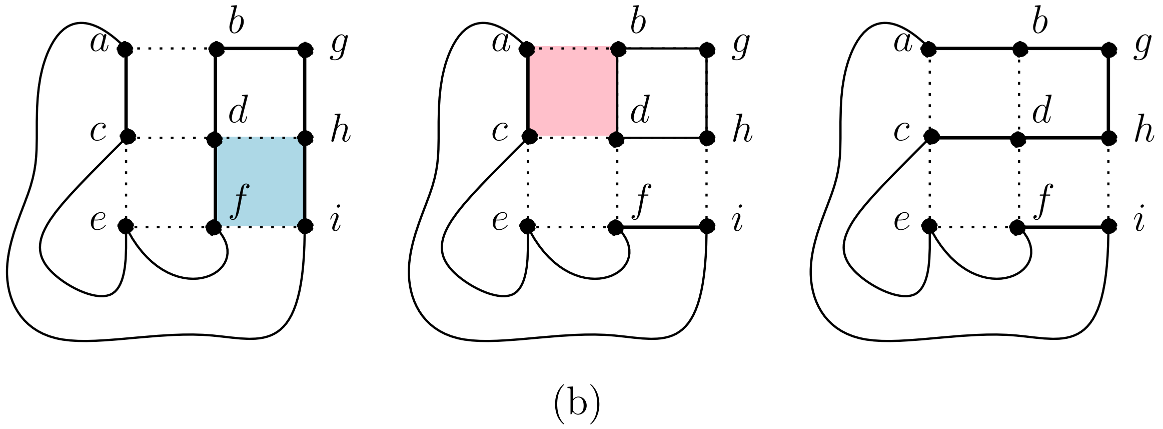


Figure 2.7: (a) A *flip* operation done by performing two *switch* operations, and (b) a *transpose* operation done by performing a pair of *switch* operations.

gives a cycle-path cover $\mathbb{P}$ of G consisting of two cycles: one cycle covers the vertices a, c, e and the other cycle covers the vertices b, d, f of G'. Then the pink cell of G' also becomes a switchable cell of G such that it contains an edge from each of the cycles of $\mathbb{P}$. The second switch is performed in the pink cell to merge the two cycles of $\mathbb{P}$ to give a Hamiltonian cycle of G.

Relationship between Transpose and Switch

Assume that G'' is a 3×3 transposable subgrid of G in C as shown in Figure 2.7(a). Then we can perform a *transpose* in G' by applying two switches. The first switch is applied in the blue cell which gives a cycle-path cover $\mathbb{P}$ of G consisting of two cycles: a cycle b, d, h, g, b of length 4 and another cycle covering all the other vertices of G. The pink cell of G' now contains an edge from each of the cycles of $\mathbb{P}$. The second switch is performed in the pink cell to merge the two cycles of $\mathbb{P}$ to give a Hamiltonian cycle of G.

2.4 Bend Complexity

Let G be a solid grid graph, and let C be a Hamiltonian cycle of G. The *bend complexity of a vertex v on C* is the minimum number of turn-free subpaths of C needed to connect v to a boundary of G. Thus the boundary vertices of G have bend complexity 0. We now introduce a new complexity measure for Hamiltonian cycles in G.

Definition 2.10 (Bend Complexity of a Cycle)**.** *The* bend complexity $B(C)$ of C *is the maximum bend complexity of any of the vertices of G on C.*

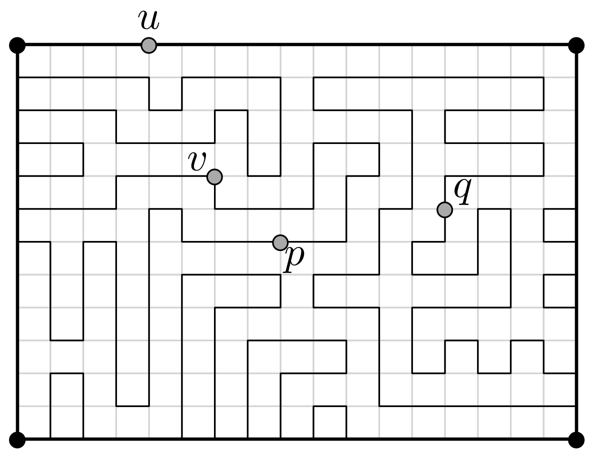


Figure 2.8: An 18-complex cycle. The vertex u on the N boundary has bend complexity 0. The internal vertices v, p and q have bend complexities $3, 11$ and 18, respectively. Note that v is a bend vertex whereas p and q are not bend vertices.

We call C a *k-complex Hamiltonian cycle* when $B(C) = k$; see Figure 2.8 for an example. Similarly we define *bend complexity for Hamiltonian paths* as follows.

Definition 2.11 (Bend Complexity of a Path)**.** *Let P be a Hamiltonian path of G. The* bend complexity $B(P)$ of P *is the maximum bend complexity of any of the vertices of G on P.*

The *cycle bend complexity $B(G)$ of a grid graph G*, or simply the *bend complexity* of G, is the maximum bend complexity $B(C)$ of any Hamiltonian cycle C of G, i.e.,

$B(G) = \max\{B(C)|C$ is a Hamiltonian cycle of $G\}$. Note that the minimum possible bend complexity, taken over all Hamiltonian cycles for G, is 1. In this section we prove an asymptotically tight linear bound $\Theta(mn)$ for $B(G)$.

The rest of this section is organized as follows. In Section 2.4.1, we show how local operations can change or preserve bend complexity of a cycle or a path. We then give upper and lower bounds on $B(G)$ in Sections 2.4.2 and 2.4.3, respectively.

2.4.1 Changing Bend Complexity with Local Operations

As we will see in Chapter 3, local operations such as *flip* can reduce or increase the bend complexity of a cycle. Take the example in Figure 2.9. The bend complexities of the two cycles C and C' are 1 and 2, respectively. C' can be obtained from C by performing the flip operation $Flip(v_{0,1}, v_{2,2})$ which increases the bend complexity of the cycle. On the other hand, a flip in the same subgrid performed on C' gives back C reducing the bend complexity of the cycle C'.

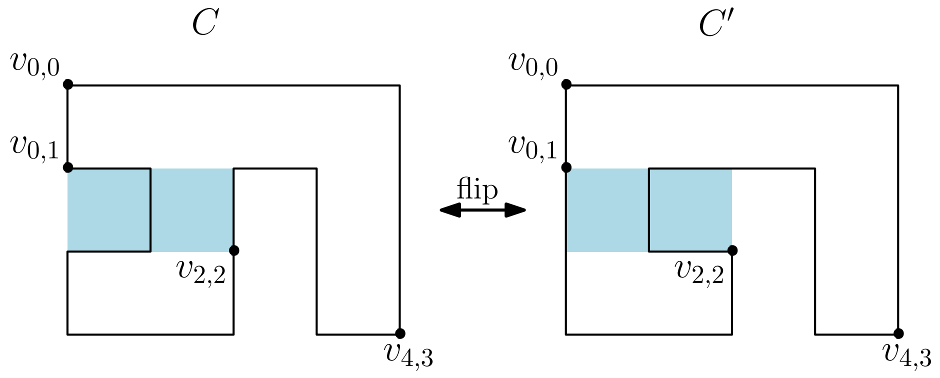


Figure 2.9: Flip operation changing bend complexity of Hamiltonian cycles C and C' between 1 and 2.

However, in Chapters 4 and 5, we will only perform flips and transposes on 1-complex Hamiltonian cycles that preserve the bend complexity, i.e., the operations we perform keep the cycle 1-complex. Figure 2.10 shows examples of such flip and transpose operations.

2.4.2 Upper Bound on $B(G)$

Let G be an $m \times n$ grid graph, where $m, n \geq 3$, and let C be a Hamiltonian cycle of G. Since any bend on C that is on the boundary of G has level 0, the bend

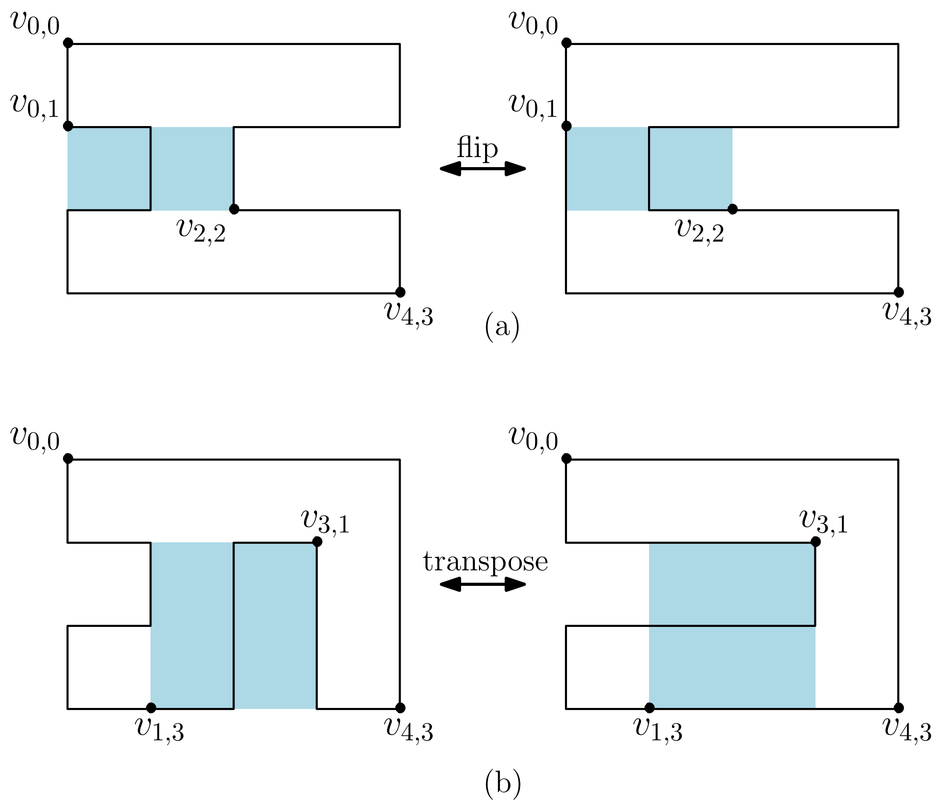


Figure 2.10: (a) Flip and (b) transpose operations preserving the bend complexity of 1-complex Hamiltonian cycles.

complexity of C arises from bends at the internal vertices (i.e., the vertices not on the boundary). There are mn vertices in total in G and $2(m+n-2)$ of them are boundary vertices. Therefore, the number of internal vertices is $mn-2(m+n-2)$. From each of the internal vertices, we can take two subpaths on the cycle to reach a boundary (N, S, E, or W). Let v be any internal vertex and let P_1 and P_2 be the two subpaths of C from v to a boundary. Then the bend complexity of v on C is the number of segments on the subpath $P \in \{P_1, P_2\}$ that has the lowest number of segments on it. To compute the highest possible bend complexity for v, assume that P_1 and P_2 cover all the internal vertices of G, and the two subpaths bend at each internal vertex of G. Then there would be at most half of the internal vertices on P. Therefore, the bend complexity of v, and hence the bend complexity $B(C)$ of C, is at most $\dfrac{mn-2(m+n-2)}{2} = \dfrac{mn}{2} - m - n + 2$. Thus $B(G) \leq \dfrac{mn}{2} - m - n + 2$.

2.4.3 Lower Bound on $B(G)$

Let G be an $m \times n$ grid graph, where we assume that m is even without loss of generality. For $m = 4$ and $n \geq 6$, we show in Chapter 3 that $B(G) = 3$. Now we assume m is even and $m \geq 6$.

We can construct a Hamiltonian cycle C with high bend complexity as shown in Figure 2.11(a), where all the vertices of G where C bends are shown as black circles and the other vertices of G are shown as white circles. Observe that a high bend complexity arises by repeating the structure in the subpath of C starting from $v_{2,m-3}$ and ending at $v_{3,m-3}$, as shown in Figure 2.11(b). Let us call each such subpath in C that starts from $v_{x',m-3}$ and ends at $v_{x'+1,m-3}$, where $(x' - 2) \mod 4 = 0$, a *tree structure*. Each tree structure except the rightmost one occupies the subgrid defined by four columns, from $x' - 1$ to $x' + 2$, and $m - 3$ rows, from 1 to $m - 3$. The rightmost tree structure occupies $4 + ((n - 2) \mod 4)$ columns, where the horizontal line segments on its right side, which begin in column x in the figure and end at column $n - 2$, are stretched to have length $1 + ((n - 2) \mod 4)$.

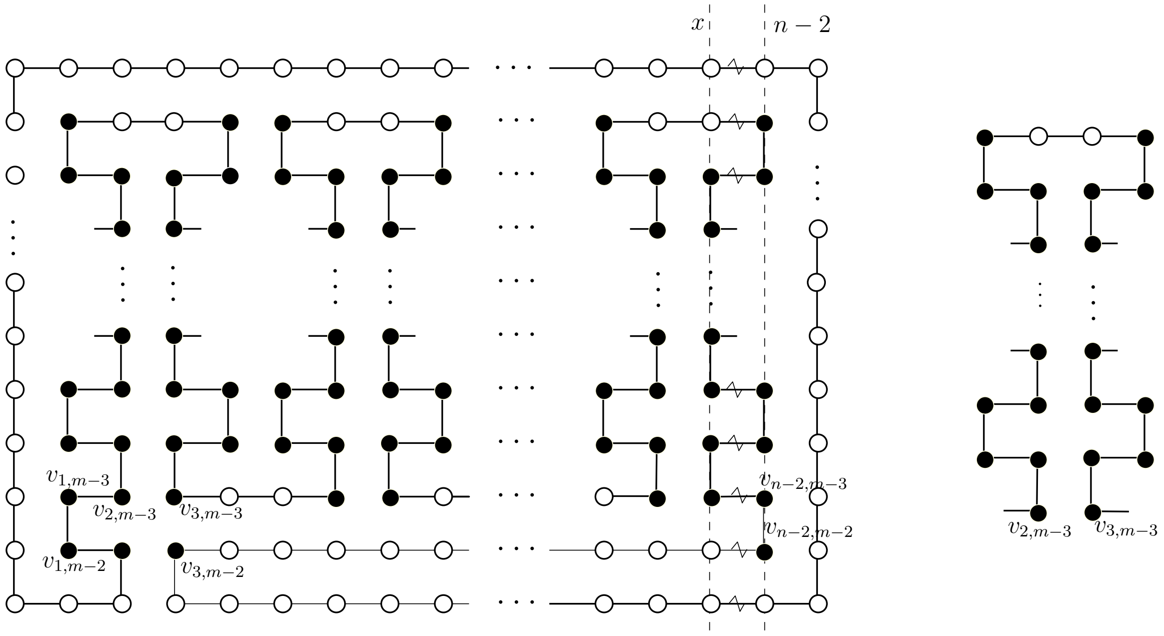


Figure 2.11: (a) A Hamiltonian cycle C with high bend complexity on an $m \times n$ grid graph, where m is even and $n \geq 6$. (b) The leftmost tree structure.

To determine the bend complexity of C, we first count the number of bends in one tree structure. From Figure 2.11(b), we see that a tree structure that occupies four columns has $m-4$ bends in each of the four columns. The rightmost tree structure, which may occupy more than four columns, will have the same number of bends. Therefore, the total number of bends in each of the tree structures is $4(m-4)$.

We now count the number of tree structures. Since each tree structure occupies at least 4 columns and none of them is incident on the east or west boundary, there are exactly $\lfloor \frac{n-2}{4} \rfloor$ tree structures.

Any internal vertex v of G can only connect to the S boundary on C, and only through two subpaths of P. One of those subpaths must go through the vertices $v_{2,m-3}, v_{1,m-3}$, and $v_{1,m-2}$; and the other subpath must go through the vertices $v_{n-2,m-3}, v_{n-2,m-2}$ and $v_{3,m-2}$. Therefore, the bend complexity $B(C)$ of C is $(\lfloor \frac{n-2}{4} \rfloor \times 4(m-4))/2+3 = \lfloor \frac{n-2}{4} \rfloor \times 2(m-4)+3$. Thus $B(G) \geq \lfloor \frac{n-2}{4} \rfloor \times 2(m-4)+3$.

2.5 Conclusion

In this chapter, we defined terminology on grid graphs that is used throughout the thesis. We also defined three local operations *flip*, *transpose* and *switch* that will later be used to reconfigure Hamiltonian paths and cycles. We introduced a new complexity measure called *bend complexity* for Hamiltonian paths and cycles in grid graphs. Based on the bend complexities of cycles in a grid graph, we defined *bend complexity* of the graph itself. Finally, we gave an asymptotically tight linear bound $\Theta(|G|)$ for the bend complexity of a rectangular grid graph G.

We conclude the chapter with some open problems.

1. The *flip* and *transpose* operations we defined in this section are equally applicable to Hamiltonian cycles and paths. Are they applicable to any paths or cycles in a grid graph? Are they applicable to cycle covers or cycle-path covers of a grid graph?

2. Can we define local operations for Hamiltonian cycles and paths in higher dimensions that are performed in a small subgrid and preserve Hamiltonicity?

3. What local operations can we define for Hamiltonian cycles or paths on other grids such as *triangular* or *hexagonal* grids that preserve Hamiltonicity?

4. The *bend complexity* measure can be applied to measure complexities of *triangular* or *hexagonal* grids. It would be interesting to achieve tight bounds for the bend

complexity for those graphs as well.

Chapter 3

Narrow Grid Graphs

We consider a restricted case of grid graphs which we use to introduce some terminology, and concepts used throughout the thesis. In particular, we introduce the notion of *cookies* and *canonical paths and cycles*. This chapter also serves to illustrate the application of the flip, transpose, and switch operations in a simple setting.

We consider $m \times n$ grid graphs for $m \leq 4$, which we call *narrow grid graphs*. Our reason for considering $m \leq 4$ is this: as we will see in Sections 3.2 and 3.3, the bend complexity of an $m \times n$ grid graph remains constant for $m \leq 4$ whereas for $m > 4$, the bend complexity can be arbitrarily large. Indeed Figure 3 shows an example of a Hamiltonian cycle C on a $5 \times n$ grid graph G where the bend complexity $B(C)$ of C is $n - 2$, and hence $B(G)$ is at least $n - 2$.

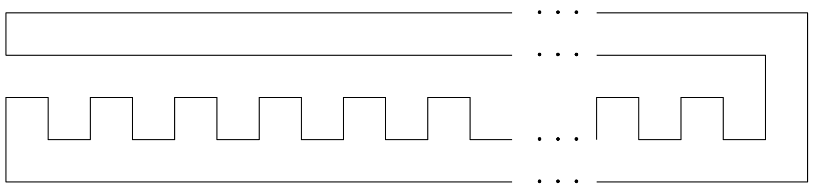


Figure 3.1: A Hamiltonian cycle C on a $5 \times n$ grid graph, where $B(C) = n - 2$.

On the other hand, in a $2 \times n$ grid graph where $n \geq 2$, there is exactly one Hamiltonian cycle as shown in Figure 3.2(a) and exactly one Hamiltonian path between the top-left corner s and the bottom-right corner t as shown in Figure 3.2(b). (Recall from Chapter 2 that throughout the rest of the thesis, unless otherwise stated, an s, t *Hamiltonian path* starts at the top-left corner and ends at the bottom-right corner of the grid graph.) Therefore, we now consider the remaining cases, namely where $m \in \{3, 4\}$, in this chapter.

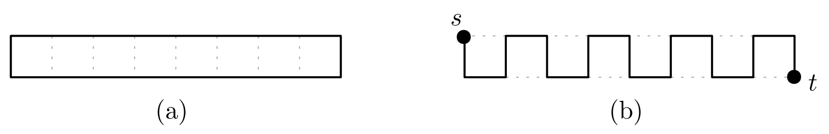


Figure 3.2: (a) The only $2 \times n$ Hamiltonian cycle, and (b) the only $2 \times n$ Hamiltonian path between the top-left and bottom-right corners.

The rest of the chapter is organized as follows.

1. In Section 3.1, we define some terminology for Hamiltonian cycles and paths that is heavily used in the rest of the chapter and also throughout the thesis. In particular we define *cookies* and *canonical paths and cycles*.

2. In Section 3.2, we show that the bend complexity of a $3 \times n$ grid graph is 1. We then give an algorithm to reconfigure any Hamiltonian *cycle* in a $3 \times n$ grid graph to any other Hamiltonian *cycle* using a linear number of flips and transposes.

3. In Section 3.3, we show that the bend complexity of a $4 \times n$ grid graph is 3. We then give an algorithm to reconfigure any Hamiltonian *cycle* in a $4 \times n$ grid graph to any other Hamiltonian *cycle* using a linear number of flips and transposes, where the bend complexities of the two cycles are not necessarily the same.

4. In Section 3.4, we give an algorithm to reconfigure any s, t Hamiltonian *path* in a $3 \times n$ grid graph to any other s, t Hamiltonian *path*, not necessarily of the same bend complexity, using a linear number of flips and transposes. Here we make use of a reconfiguration algorithm for $4 \times n$ cycles, a technique that we repeat to reconfigure s, t Hamiltonian paths later in Chapter 6.

5. In Section 3.5, we show that the complexity of an s, t Hamiltonian path in a $4 \times n$ grid can be arbitrarily large while a cycle in the same grid has constant bend complexity. We then give an algorithm to reconfigure any s, t Hamiltonian *path* in a $4 \times n$ grid graph to any other s, t Hamiltonian *path* using a linear number of flips, transposes and switches. This algorithm illustrates the application of *pairs of switches* that preserve Hamiltonicity, as we will see again in Chapter 6.

6. Section 3.6 concludes with some open problems.

3.1 Cookies and Canonical Cycles/Paths

Let C be a Hamiltonian cycle in an $m \times n$ grid graph G, and let P be an s, t Hamiltonian path of G. To avoid trivialities, we assume $m > 2$.

Definition 3.1 (Cookie)**.** *A* cookie *c is a sequence of consecutive vertices on C or P that begins and ends on adjacent non-corner vertices on a boundary of G and that does not contain any other vertices on the boundaries of G.*

Refer to Figure 2.11 in Chapter 2 for an example of a cookie in a general case for arbitrary m. In that example, all the internal vertices of the grid graph is covered by only one cookie.

We observe some properties of C in terms of cookies.

Lemma 3.1. *Every internal vertex v of G must be on some cookie of C.*

Proof. Follow C in both directions from v and let a and b be the first boundary vertices encountered. Neither a nor b can be a corner vertex. If they are not adjacent on the same boundary of G then there are some boundary vertices that cannot be covered by C, contradicting the Hamiltonicity of C. Therefore, a and b must be the endpoints of a cookie of C that covers v. □

Lemma 3.2 (Structure of C)**.** *C consists of cookies, together with sequences of vertices on C that form straight line segments on the boundary connecting two cookies, or a cookie to a corner vertex, or two corner vertices on the same boundary.*

Proof. The proof follows directly from Lemma 3.1. □

We remark that the above definitions and lemmas will be repeated in Chapter 5 where we consider L-shaped grid graphs. Indeed, these definitions and lemmas apply to any solid grid graphs (without holes).

The *base* of a cookie is the boundary edge of G that connects the two endpoints of the cookie. Based on which boundary the base of a cookie lies, we have *four types* of cookies: W,E,N, and S.

We now define special kinds of 1-complex cycles and 1-complex paths that we call, respectively, *canonical Hamiltonian cycles* and *canonical s, t Hamiltonian paths.* These canonical cycles and paths will be used as intermediate configurations in our reconfiguration algorithms. In more general settings, other definitions of canonical paths and cycles can be considered.

Definition 3.2 (Rectangular canonical Hamiltonian cycles)**.** *A* rectangular canonical Hamiltonian cycle *is a* 1*-complex cycle in an $m \times n$ rectangular grid graph G such that the cycle has exactly one type of cookie (N,S,E, or W).*

If both m and n are even, then G has four canonical Hamiltonian cycles. Otherwise G has two canonical Hamiltonian cycles when one of m, n is even and zero Hamiltonian cycles when both are odd.

Definition 3.3 (Canonical s, t Hamiltonian paths)**.** *A* canonical s, t Hamiltonian path *is a* 1*-complex path in an $m \times n$ rectangular grid graph G that bends only on the boundaries of G.*

If both m and n are odd, then G has two canonical Hamiltonian paths; otherwise, G has one canonical Hamiltonian path when one of m, n is odd, and zero Hamiltonian paths when both are even.

3.2 $3 \times n$ Hamiltonian Cycles

The $m = 3$ case is the first non-trivial case of a narrow grid graph. Let G be a $3 \times n$ grid graph. We assume n is even as otherwise, G has no Hamiltonian cycle. We first establish a structure for any Hamiltonian cycle in G.

Lemma 3.3 (Structure of $3 \times n$ Hamiltonian cycles)**.** *Let C be a Hamiltonian cycle of a $3 \times n$ grid graph G, where $n \geq 4$ is even. Then C consists of only N or S cookies, but no W or E cookies, such that each N or S cookie is contained in a track t^v_x, where x is odd. The interiors of tracks $t^v_{x'}$, where x' is even, do not intersect any cookies.*

Proof. Any vertex of G must have two incident edges in C. Since the four corner vertices of G have degree two, they have both their incident edges in C. Therefore, there are no W or E cookies in C.

Since each internal vertex of G must be on a cookie of C by Lemma 3.1, they must either be on an N or S cookie. Assume that the vertex $v_{1,1}$ is on an N cookie c. Thus c has $v_{1,0}$ on the N boundary as an endpoint and $v_{2,0}$ as the other endpoint. Therefore, vertex $v_{2,1}$ and edge $(v_{2,0}, v_{2,1})$ lie on cookie c. Since the only path internal to the grid connecting $v_{1,1}$ to $v_{2,1}$ is the horizontal edge between those vertices, that edge must lie on c which completes the cookie, which lies in track t^v_1. The case for $v_{1,1}$ on an S cookie is similar. In either case, track t^v_2 has an empty interior. In this

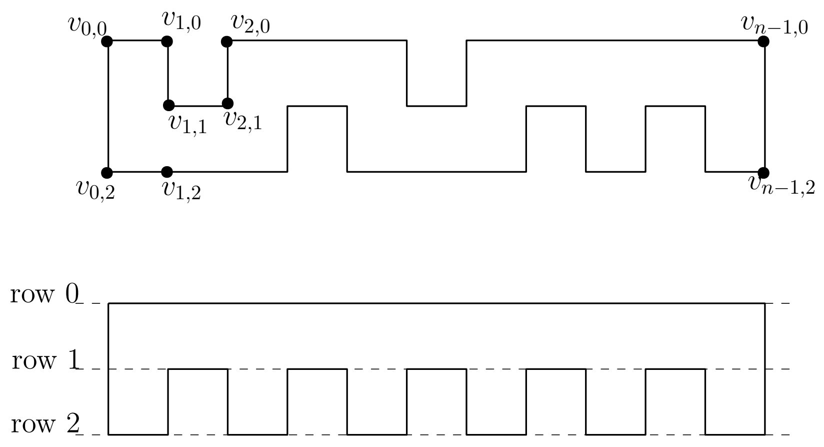


Figure 3.3: (a) A Hamiltonian cycle, and (b) a canonical Hamiltonian cycle.

way, every alternate edge in Row 1 starting from $(v_{1,1}, v_{2,1})$ must be on C, and the endpoints of each such edge must be connected either to the N or the S boundary in C, forming either an N or an S cookie, respectively, in tracks t^h_x, where x is odd. □

We now show that any Hamiltonian cycle on G has bend complexity 1, and thus $B(G) = 1$.

Lemma 3.4. *Let G be a $3 \times n$ grid graph and let C be any Hamiltonian cycle of G. Then the following hold.*

(a) There are exactly $\frac{n-2}{2}$ cookies in C.

(b) $B(C) = 1$.

(c) $B(G) = 1$.

(d) There are $2^{\frac{n-2}{2}}$ Hamiltonian cycles of G.

Proof. By Lemma 3.3, each t^v_x contains an N or S cookie, where x is odd. Since there are $\frac{n-2}{2}$ such tracks, there are exactly $\frac{n-2}{2}$ cookies in C, proving Claim (a).

Since each internal vertex of G is connected to the N or S boundary by an edge, they all have bend complexity 1, proving Claims (b) and (c).

For each cookie of C, we have two choices: it can be either an N or an S cookie. So there are $2^{\frac{n-2}{2}}$ Hamiltonian cycles of G, proving Claim (d). □

We now show that for $3 \times n$ grid graphs, we can reconfigure Hamiltonian cycles using only a linear number of *flips.*

Theorem 3.1. *Let G be a $3 \times n$ grid graph and let C_1 and C_2 be any two Hamiltonian cycles of G. Then C_1 can be reconfigured to C_2 using at most $\frac{n-2}{2}$ flip operations.*

Proof. For each track t_x^v, where x is odd, if C_1 and C_2 have the same type of cookie (N or S) in that track, do nothing. Otherwise, apply $Flip(v_{x,0}, v_{x+1,2})$ on C_1. Since there are exactly $\frac{n-2}{2}$ cookies in C_1 by Lemma 3.4, at most $\frac{n-2}{2}$ flips are required. $\square$

3.3 $4 \times n$ Hamiltonian Cycles

In this section, we assume $n \geq 4$ as the 4×3 cycles are covered by the previous section. We define the shapes shown in Figure 3.4 for cookies in a Hamiltonian cycle C in a $4 \times n$ grid. The base (p, q) of an I shaped cookie lies on some boundary N, S, E or W, of G; the bases of the other shapes lie on the N or S boundary. An E-facing

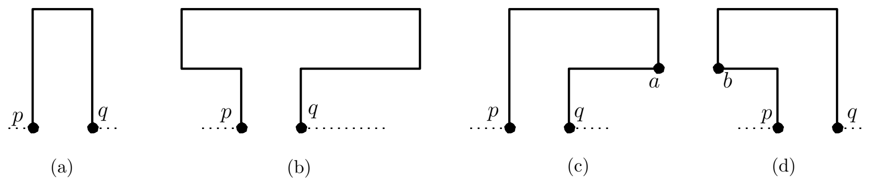


Figure 3.4: The four shapes for cookies in a $4 \times n$ Hamiltonian cycle: (a) I, (b) T, (c) E-facing (i.e., $x(a) > x(q)$), and (d) W-facing (i.e., $x(b) < x(p)$).

(W-facing) cookie has a vertical edge that is closer to the E (W) boundary than the base of the cookie, which lies either on the N or S boundary.

We now show that these four shapes are the only shapes any cookie of C can have.

Lemma 3.5. *Let C be a Hamiltonian cycle of a $4 \times n$ grid graph G. Then any cookie of C must have one of the four shapes: I, T, E-facing, and W-facing.*

Proof. It is straightforward to see that there can be at most one E cookie and at most one W cookie in C. Since there are only four rows and the vertices of an E or a W cookie can only be in horizontal track t_1, such a cookie must be I shaped. On the other hand, an N or an S cookie can have any of the four shapes shown in Figure 3.4. For $m = 4$, we enumerate them by applying the following logic. By Lemma 3.1 each internal vertex must be on some cookie of C. Assume that edge $v_{x,3}, v_{x+1,3}$ of G is the base of an S cookie c as shown in Figure 3.5. Then the vertical edges $(v_{x,3}, v_{x,2})$ and $(v_{x+1,3}, v_{x+1,2})$ must be in c.

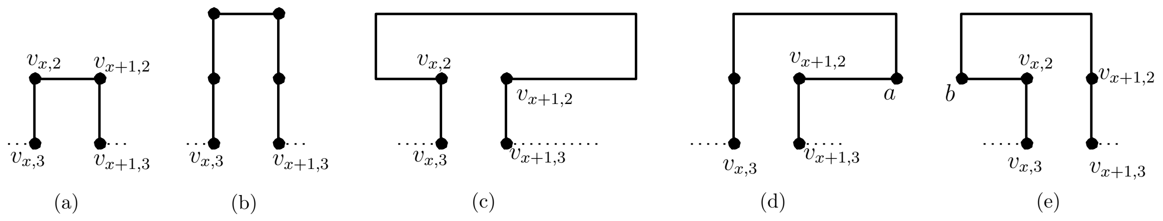


Figure 3.5: An S cookie c with the base $(v_{x,3}, v_{x+1,3})$: (a) c does not intersect Row 1, (b) cookie c intersects Row 1 but does not have bends in Row 2, (c) cookie c intersects Row 1 and has two bends on Row 2, (d) in Row 2, cookie c bends only at $v_{x+1,2}$ but not at $v_{x,2}$, and (e) in Row 2, cookie c bends only at $v_{x,2}$ but not at $v_{x+1,2}$.

If c intersects only one internal row, namely Row 2, then the horizontal edge $(v_{x,2}, v_{x+1,2})$ completes the cookie and it is I shaped (see Figure 3.5(a)). Otherwise c intersects both Rows 1 and 2. Based on whether the vertices $v_{x,2}$ and $v_{x+1,2}$ have bends, we have four different shapes for c: if neither of the vertices has a bend c is an I shaped cookie (see Figure 3.5(b)); if both vertices have bends c is a T shaped cookie (see Figure 3.5(c)); if only $v_{x+1,2}$ has a bend c is an E-facing cookie (see Figure 3.5(d)); and if only $v_{x,2}$ has a bend c is a W-facing S cookie (see Figure 3.5(e)). Similarly, the four different shapes an N cookie with base $(v_{x,0}, v_{x+1,0})$ can take are shown in Figures 3.6(a)–(e). □

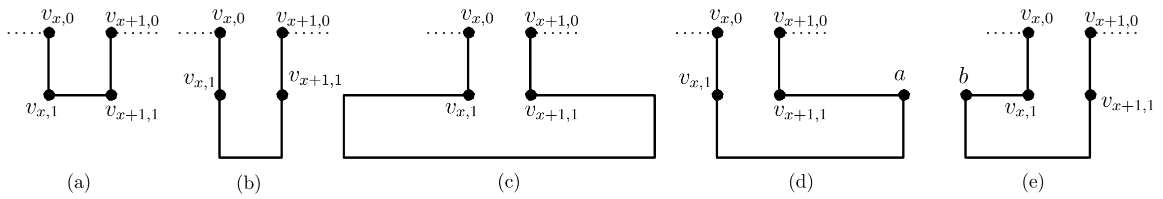


Figure 3.6: An N cookie with base $(v_{x,0}, v_{x+1,0})$: (a)–(b) I shaped, (c) T shaped, (d) E-facing $(x(a) > x + 1)$, and (e) W-facing $(x(b) < x)$.

We call an I shaped N or S cookie that intersects only one internal row a *short* cookie.

Lemma 3.6. *If a vertical track contains a short cookie, then it must also contain another short cookie from the opposite boundary.*

Proof. Suppose the statement is not true. Let t_x^v be the westmost track for which the statement is not true. Suppose t_x^v contains only a short N cookie. Then the vertex

$v_{x,2}$ must be covered by a short S cookie contained in track t^v_{x-1}, as it cannot be covered by a W or E cookie. This is a contradiction to t^v_x being the westmost track containing exactly one short cookie. The case of a vertical track containing only an S cookie is similar. □

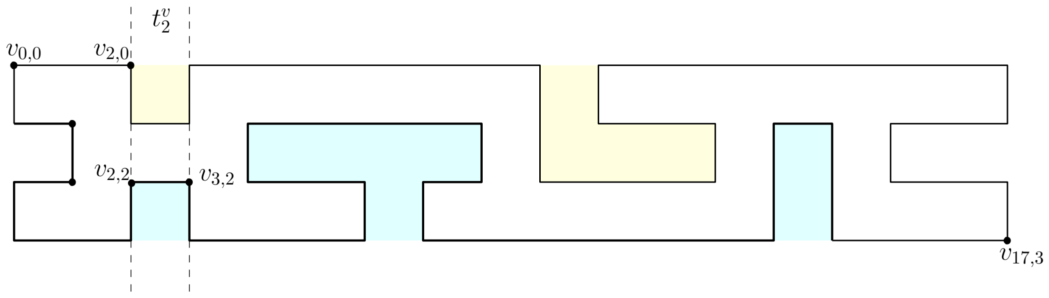


Figure 3.7: A $4 \times n$ Hamiltonian cycle, where t^v_2 contains both an I shaped N and an I shaped S cookie.

We now compute the bend complexity $B(G)$ of G.

Lemma 3.7. *Let G be a $4 \times n$ grid graph and let C be any Hamiltonian cycle of G. Then $B(C) \leq 4$. Moreover, $B(G) = 1$ for $n = 4$, $B(G) = 2$ for $n = 5$, and $B(G) = 4$ for $n \geq 6$.*

Proof. By Lemma 3.5, any cookie of C must have one of the four shapes: I, T, E-facing, and W-facing, and by Lemma 3.1 every internal vertex of G must be on a cookie of C. The bend complexity of any internal vertex is exactly 1 if it is on an I shaped cookie, at most 2 if it is on an E-facing or W-facing cookie, and at most 4 if it is on a T shaped cookie. Therefore, the bend complexity $B(C) \leq 4$. If G is a 4×4 grid graph, C can only have I shaped cookies and therefore, $B(G) = 1$. If G is a 4×5 graph, C can have an E-facing or W-facing cookie but no T shaped cookie. Therefore $B(G) = 2$ for $n = 5$. When $n \geq 6$, C can have cookies of any of the four shapes and therefore $B(G) = 4$. □

The proof of the next theorem is constructive and gives an algorithm to reconfigure any Hamiltonian cycle of G to a canonical form.

Theorem 3.2. *Let C be a Hamiltonian cycle in a $4 \times n$ grid graph G. Then C can be reconfigured to any other Hamiltonian cycle C' of G using at most $4n$ flips and transposes.*

Proof. Let $\mathbb{C}$ be the canonical Hamiltonian cycle of G with a W cookie. We show that each of C and C' can be reconfigured to $\mathbb{C}$ using $2n$ flips and transposes. Hence, by reversing the steps to reconfigure C' to $\mathbb{C}$, we can transform C to C' via $\mathbb{C}$ using at most $4n$ flips and transposes.

Step 1. Let t^v_a be a track containing a short cookie. Then it must contain another short cookie from the opposite boundary by Lemma 3.6. We apply $Flip(v_{a,0}, v_{a+1,2})$ in each t^v_a, $1 \leq a \leq n-3$, so that after the flip the track t^v_a contains only an I shaped S cookie intersecting Rows 1 and 2 (see Figure 3.8(a)). Let the cycle obtained at the

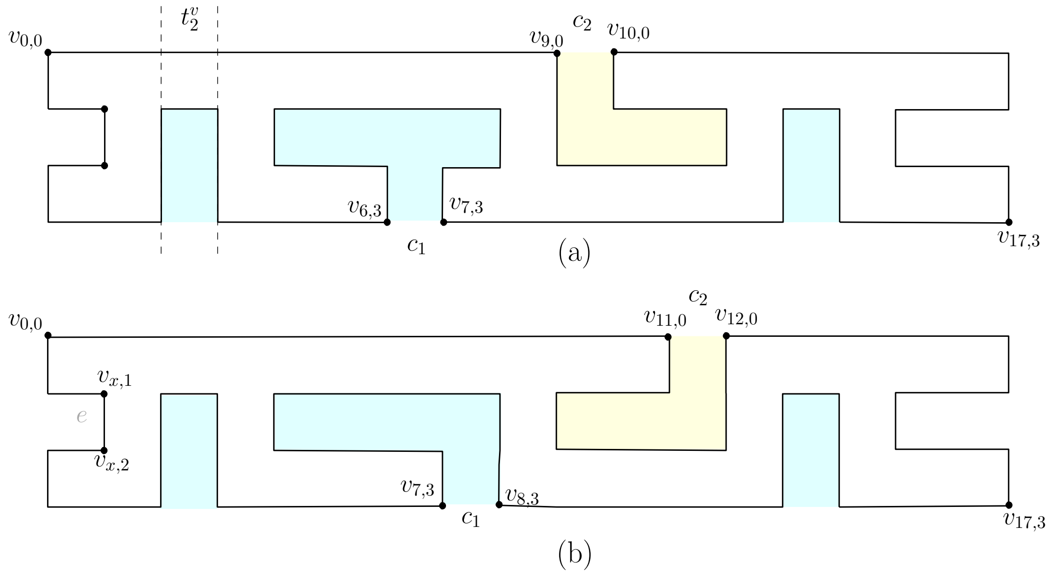


Figure 3.8: (a) After applying flip in track t^v_2 on the Hamiltonian cycle from Figure 3.7. (b) After applying flips to make c_1 and c_2 W-facing. Note that the bases of both the cookies have moved to the right.

end of Step 1 be C_1. If C_1 does not have any N or S cookies go to Step 4. Otherwise, go to Step 2.

Step 2. For each N or S cookie of C_1 that is T shaped or E-facing, make it W-facing by applying flips and moving the base of the cookie to the east as follows. Let c be an N cookie with base $(v_{p,0}, v_{p+1,0})$. Then we do the following as long as vertex $v_{p+1,1}$ has a bend (that means c is either T shaped or E-facing): apply $Flip(v_{p,0}, v_{p+2,1})$ and set $p = p + 1$.

We repeat this process for all the N and S cookies of C_1 to obtain a Hamiltonian cycle that has only W-facing and I shaped N or S cookies that all intersect both internal rows (see Figure 3.8(b)). We then move to Step 3.

Step 3. Let C_2 be the cycle obtained at the end of Step 2. If C_2 has a W cookie let $e = (v_{x,1}, v_{x,2})$ be the only vertical edge of the W cookie; otherwise, C_2 does not have a W cookie and we let $e = v_{0,1}, v_{0,2}$. Let c be the N or S cookie of C_2 that is closest to the W boundary (c changes as the algorithm progresses and track t_1^h gradually fills with a W cookie). We consider the following cases.

1. If c is an I shaped S cookie, we apply $Xpose(v_{x,3}, v_{x+2,1})$ and set $x = x + 2$.

2. If c is an I shaped N cookie, we apply $Xpose(v_{x,0}, v_{x+2,2})$ and set $x = x + 2$.

3. Otherwise, c must be a W-facing cookie. Then we apply $Flip(v_{x,1}, v_{x+2,2})$ and set $x = x + 1$.

We continue Step 3 until there are no N or S cookies. If we have just one W cookie covering all the internal vertices, then it is the canonical Hamiltonian cycle $\mathbb{C}$. Otherwise, we move to Step 4.

Step 4. Let C_3 be the cycle obtained at the end of Step 3. If C_3 has a W cookie let $e = (v_{x,1}, v_{x,2})$ be the only vertical edge of the W cookie; otherwise, C_3 does not have a W cookie and $e = v_{0,1}, v_{0,2}$. Apply $Flip(v_{x,1}, v_{x+2,2})$ and set $x = x + 1$ until $x = n - 2$. At that point we have obtained $\mathbb{C}$.

Steps 1 and 2 take at most n flip operations together, since Step 1 applies flips on I shaped cookies which then remain untouched in Step 2. Steps 3 and 4 take at most n flip and transpose operations in total. So the total number of operations needed in reconfiguring C to $\mathbb{C}$ is at most $2n$. Hence the reconfiguration of C to C' takes at most $4n$ operations.

□

Remark Figure 3.9 shows an example where the algorithm implicit in the proof of Theorem 3.2 requires exactly $2n - 6$ flips and 1 transpose to reconfigure each of C_1 and C_2 to $\mathbb{C}$, which implies an upper bound of $\theta(n)$ on the number of local operations used by the algorithm implicit in the proof of Theorem 3.2.

Note that for this particular example, it is possible to find a more efficient way to reconfigure with flips and transposes. For example, C_1 can be reconfigured to C_2 using only two transposes: $Xpose(v_{2,1}, v_{0,3})$ and then $Xpose(v_{2,2}, v_{0,0})$.

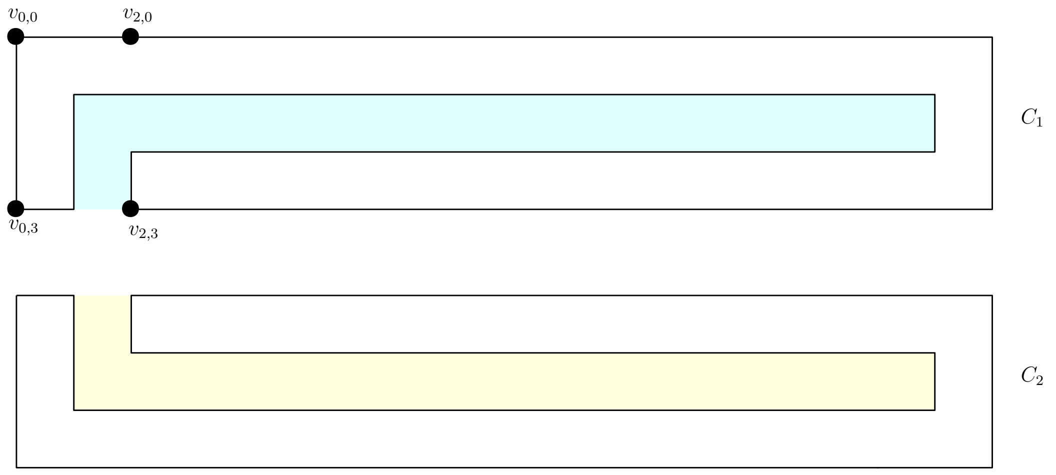


Figure 3.9: Two Hamiltonian cycles C_1 and C_2 in a $4 \times n$ grid graph.

3.4 $3 \times n$ Hamiltonian Paths

Let P be a Hamiltonian path from the top-left corner s to the bottom-right corner t of a $3 \times n$ grid graph G as shown in Figure 3.10(a). We have the following lemma.

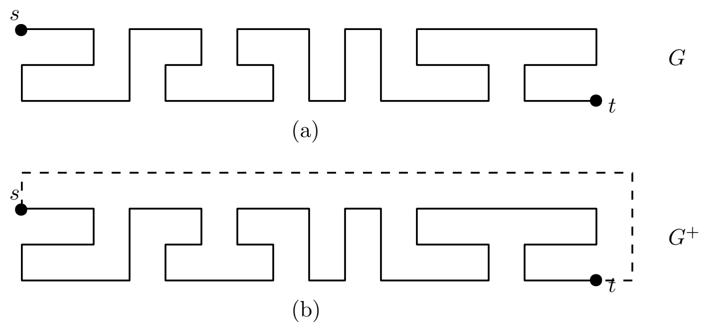


Figure 3.10: (a) s, t Hamiltonian path P of G, (b) Hamiltonian cycle C of G^+.

Lemma 3.8. *Let G be a $3 \times n$ grid graph and let P be any s, t Hamiltonian path of G. Then $B(P) \leq 2$.*

Proof. We augment G by adding a row above the N boundary of G and then a column to the right of the E boundary to obtain the $4 \times (n + 1)$ grid graph G^+. We then connect s to t through the augmented vertices (see Figure 3.10(b)) to obtain a Hamiltonian cycle C in G^+ such that C has no E or N cookies. By Lemma 3.7, $B(C) \leq 4$, and any cookie of C with bend complexity more than one must be based in the N or S boundary and has one of the four shapes (I, T, E-facing and W-facing). Clearly in this case such a cookie is an S cookie. All the internal vertices of G are also internal to G^+ and hence by Lemma 3.1 must lie on a cookie of cycle C of G^+, in this case, a W cookie or an S cookie of one of the four shapes. The statement of the lemma now follows by inspection. □

We now have the following reconfiguration result.

Theorem 3.3. *Let P_1 and P_2 be any two s, t Hamiltonian paths in a $3 \times n$ grid graph G. Then P_1 can be reconfigured to P_2 using at most $4n$ flips and transposes.*

Proof. Let C_1 and C_2 be the two $4 \times n$ Hamiltonian cycles obtained from P_1 and P_2, respectively, as in the proof of Lemma 3.8. Note that both C_1 and C_2 have no N or E cookies and any S cookie of C_1 and C_2 intersects both the internal rows of G^+. By Theorem 3.2, C_1 can be reconfigured to C_2 using $4n$ flips and transposes. Since neither C_1 nor C_2 has any short cookies, Step 1 of the algorithm implicit in the proof of Theorem 3.2 is not applied. Similarly, Step 4 of that algorithm is not applied since neither C_1 nor C_2 has any E cookies. Since C_1 and C_2 have no N cookies, all the flip operations in Step 2 of the algorithm are applied in 2×3 subgrids of G^+ contained in Rows 1 and 2 of G^+, which are the same as Rows 0 and 1 of G; and all the transpose operations in Step 3 of the algorithm are applied in 3×3 subgrids of G^+ contained in Rows 1–3 of G^+, which are the same as Rows 0–2 of G. Therefore, we can apply the sequence of flips and transposes applied to reconfigure C_1 to C_2 by the algorithm implicit in the proof of Theorem 3.2, namely, we apply these operations on G to reconfigure P_1 to P_2 by adjusting the row numbers, i.e., substituting Row x, $1 \leq x \leq 3$, of G^+ by Row $x - 1$ of G. By Theorem 3.2, at most $4n$ flips and transposes are required for the reconfiguration. □

3.5 $4 \times n$ Hamiltonian Paths

The $4 \times n$ grid graph is interesting because it has constant cycle bend complexity, i.e., $B(C) \leq 4$ by Lemma 3.7, but there are $4 \times n$ s, t Hamiltonian paths with arbitrarily large bend complexity. Remember from the introduction of the chapter that the upper bound for the bend complexity of $5 \times n$ Hamiltonian cycles is $\Theta(n)$. The family of $4 \times n$ s, t Hamiltonian paths illustrated in Figure 3.11, similar to the family of $5 \times n$ Hamiltonian cycles in Figure 3, has bend complexity $\Theta(n)$.

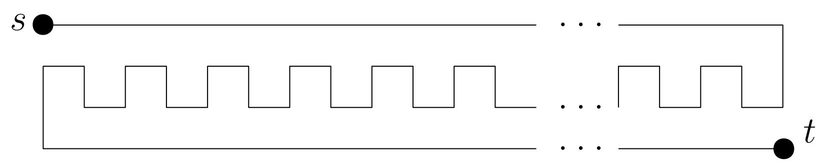


Figure 3.11: An s, t Hamiltonian path on a $4 \times n$ grid graph with bend complexity $\Theta(n)$.

Let P_1 and P_2 be two s, t Hamiltonian paths on a $4 \times n$ grid graph G, where n is odd. Let $\mathbb{P}$ be the canonical Hamiltonian path of G as shown in Figure 3.12. Since 4 is even, this is the only possible canonical form consistent with Definition 3.3. In this section, we give an algorithm to reconfigure P_1 to P_2 via $\mathbb{P}$. As in the previous sections, we reconfigure both P_1 and P_2 to $\mathbb{P}$, and then reverse the steps to reconfigure $\mathbb{P}$ to P_2. Therefore, it suffices to show that P_1 can be reconfigured to $\mathbb{P}$ using a linear number of flips, transposes and *switches*. Here we use the switch operation for the first time.

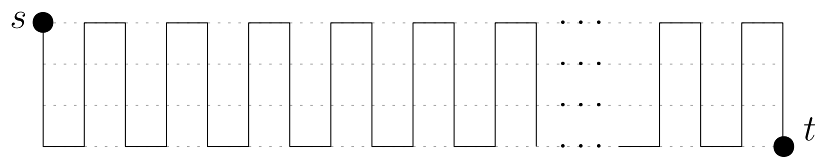


Figure 3.12: The canonical Hamiltonian path on a $4 \times n$ grid graph, n is odd.

Theorem 3.4. *Any s, t Hamiltonian path P in a $4 \times n$ grid graph can be reconfigured to $\mathbb{P}$ using at most $3(n-1)/2$ transpose, flip and switch operations.*

Proof. We look at the rightmost 4×3 subgraph of G, which we call the *window*. We do a straightforward case analysis based on how many of the edges of Column $n-3$, namely edges $(v_{n-3,0}, v_{n-3,1})$, $(v_{n-3,1}, v_{n-3,2})$ and $(v_{n-3,2}, v_{n-3,3})$, are on the Hamiltonian path P. The case analysis leads to the 16 possible patterns as shown in Figures 3.16 and 3.17. The colored cells indicate the subgrid where we later apply local operations to reconfigure path P to the form shown in Figure 3.13.

We give an example to show how we obtain the 16 patterns. Suppose only the edge $(v_{n-3,0}, v_{n-3,1})$ of Column $n-3$ is on P. Then the edges $(v_{n-3,2}, v_{n-2,2})$ and $(v_{n-3,3}, v_{n-2,3})$ must also be on P. Since every vertex of G other than s and t must contribute two edges to P, and the vertex $v_{n-1,0}$ has degree two in G, both of its incident edges must be on P. The vertex $t = v_{n-1,3}$ must have exactly one of its incident edges on P.

First we suppose that the edge $(v_{n-2,3}, t)$ on the S boundary is on P. Then the vertex $v_{n-1,2}$ just above t cannot connect to t on P and therefore, we must have the edges $(v_{n-2,2}, v_{n-1,2})$ and $(v_{n-1,1}, v_{n-1,2})$ on P. Now consider the vertex $v_{n-2,1}$. Since two of its neighbors, $v_{n-2,2}$ and $v_{n-1,1}$, have degree two on P at this point, the edges $(v_{n-3,1}, v_{n-2,1})$ and $(v_{n-2,0}, v_{n-2,1})$ incident to $v_{n-2,1}$ must be on P. Thus we obtain Pattern 1.

Now we assume that the vertical edge $(v_{n-1,2}, t)$ is on P. Then the vertex $v_{n-2,3}$ cannot have an edge with t on P and the edge $(v_{n-2,2}, v_{n-2,3})$ must be on P. The edge $(v_{n-1,1}, v_{n-1,2})$ must be on P since $v_{n-1,2}$ cannot connect to $v_{n-2,2}$ at this point. For vertex $v_{n-2,1}$, the only neighbors that do not have degree 2 on P are $v_{n-3,1}$ and $v_{n-2,0}$. Hence the edges connecting $v_{n-2,1}$ to those two neighbors must be on P, resulting in Pattern 2.

It is straightforward to enumerate the other patterns using similar reasoning.

We now sketch the straightforward proof that for each of the 16 patterns, at most 3 flips, transposes and switches reconfigure P to another s, t Hamiltonian path P' such that P' has the following form (see Figure 3.13): all the vertical edges in Columns $n-2$ and $n-1$ must be on P', the edges $(v_{n-2,0}, v_{n-1,0})$ and $(v_{n-3,3}, v_{n-2,3})$ must be on P', and each of the vertical edges of Column $n-3$ may or may not be on P'. Consequentially, the horizontal edges $(v_{n-3,0}, v_{n-2,0})$, $(v_{n-3,1}, v_{n-2,1})$, and $(v_{n-3,2}, v_{n-2,2})$ between Columns $n-3$ and $n-2$, and the horizontal edges $(v_{n-2,1}, v_{n-1,1})$, $(v_{n-2,2}, v_{n-1,2})$, and $(v_{n-2,3}, v_{n-1,3})$ between Columns $n-2$ and $n-1$ of G must not be on P'.

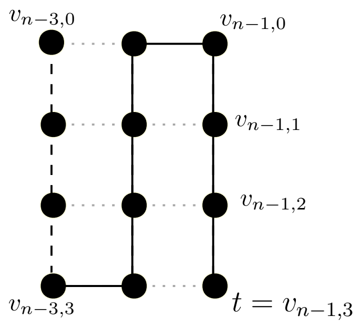


Figure 3.13: In the window covering the last three columns of G: P' must contain the edges of G shown in black and must not contain the dotted edges; the dashed edges may or may not be on P'.

We describe the operations on the first two patterns in detail as an illustration.

Pattern 1. We apply a switch in the switchable cell with the vertices $v_{n-3,2}$ and $v_{n-2,1}$, shown in pink in Figure 3.14(a). By Lemma 2.3, this switch produces a cycle-path cover $\mathbb{P}$ containing a cycle C covering the 3×2 subgrid defined by the intersection of Columns $n-2$ and $n-1$ of G and the Rows 0–2 of G, and a path P'' from s to t covering the rest of the vertices of G. We then apply another switch in the switchable cell f with the vertices $v_{n-2,2}$ and t, shown in blue in Figure 3.14(b). Since one of the edges of the blue cell belonged to C and the other belonged to P'' before the switch, by Lemma 2.4, this operation gives an s, t Hamiltonian path P' of G; see Figure 3.14(c).

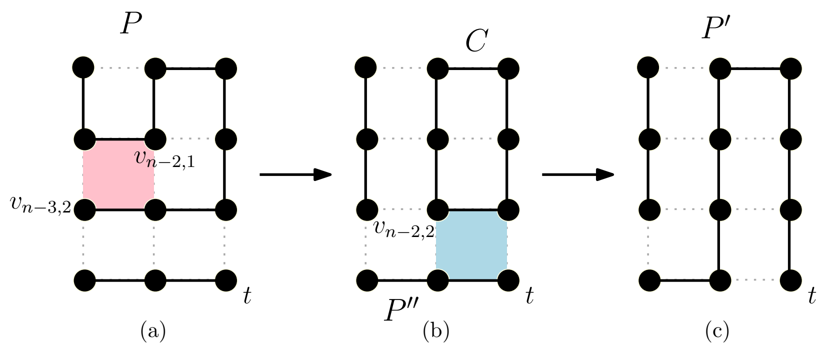


Figure 3.14: (a) An s, t Hamiltonian path P of G. (b) A cycle-path cover $\mathbb{P}$ of G with a cycle C and a path P'' from s to t, obtained from P by applying a switch in the pink cell. (c) An s, t Hamiltonian path P' of G, obtained from $\mathbb{P}$ by applying a switch in the blue cell.

Pattern 2. For this pattern, we widen our window by one column to the left as shown in Figure 3.15(a). Then the horizontal edge $(v_{n-4,0}, v_{n-3,0})$ must be on P since vertex $v_{n-3,0}$ is not a terminal vertex of P and hence must contribute two edges to P. Since the vertical edges $(v_{n-3,1}, v_{n-3,2})$ and $(v_{n-3,2}, v_{n-3,3})$ cannot be on P in Pattern 2, and the non-terminal vertices $v_{n-3,2}$ and $v_{n-3,3}$ must have two edges on P, the horizontal edges $(v_{n-4,2}, v_{n-3,2})$ and $(v_{n-4,3}, v_{n-3,3})$ must be on P. We then apply $Xpose(v_{n-4,1}, v_{n-2}, 3)$ to obtain P' as shown in Figure 3.15(b), since by Lemma 2.2, a transpose operation preserves Hamiltonicity.

Pattern 3. We apply $Flip(v_{n-3,0}, v_{n-2,2})$ to get P'' which has Pattern 1 in the window. On P'' we then apply the switch operations we applied in the case of

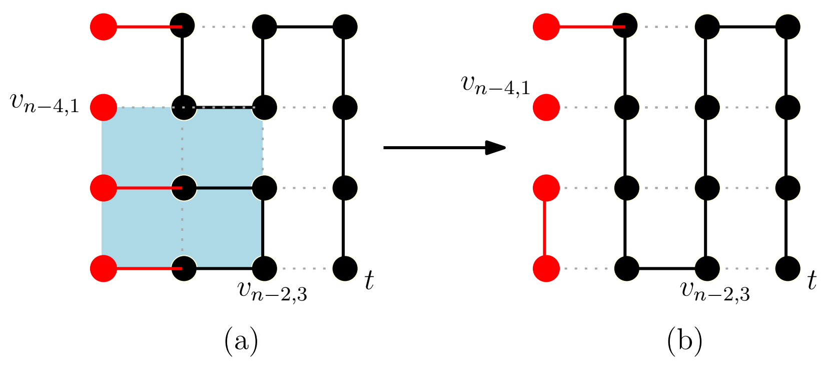


Figure 3.15: (a) An s, t Hamiltonian path P of G. (b) An s, t Hamiltonian path P' of G, obtained from P by applying transpose in the blue 3×3 subgrid with the top left corner of the subgrid as the pin vertex.

Pattern 1 to obtain P'.

Pattern 4. We apply $Flip(v_{n-3,0}, v_{n-2,2})$ to get P'' which has Pattern 2 in the window. On P'' we then apply the switch operations we applied in the case of Pattern 2 to obtain P'.

Pattern 5. We apply the transpose operation $Xpose(v_{n-3,2}, v_{n-1,0})$.

Pattern 6. Apply a switch in the pink cell that converts P to a cycle-path cover $\mathbb{P}$ consisting a cycle and a path. Then apply a switch in the blue cell on the cycle-path cover $\mathbb{P}$ to obtain P'.

Pattern 7. Apply the transpose operation $Xpose(v_{n-3,2}, v_{n-1,0})$.

Pattern 8. For this pattern, we widen our window by one column to the left. We apply a switch in the pink cell which converts P to a cycle-path cover $\mathbb{P}$ consisting of a cycle and a path. We then apply a switch in the blue cell on $\mathbb{P}$ to obtain P'.

Pattern 9. Apply a switch in the pink cell followed by a switch in the blue cell.

Pattern 10. We apply the transpose operation $Xpose(v_{n-3,2}, v_{n-1,0})$.

Pattern 11. Apply the transpose operation $Xpose(v_{n-1,1}, v_{n-3,3})$.

Pattern 12. Apply the flip operation $Flip(v_{n-3,2}, v_{n-1,3})$ to obtain an s, t Hamiltonian path P'' which has Pattern 2 in the window. On P'' we then apply the switch operations we applied in the case of Pattern 2 to obtain P'.

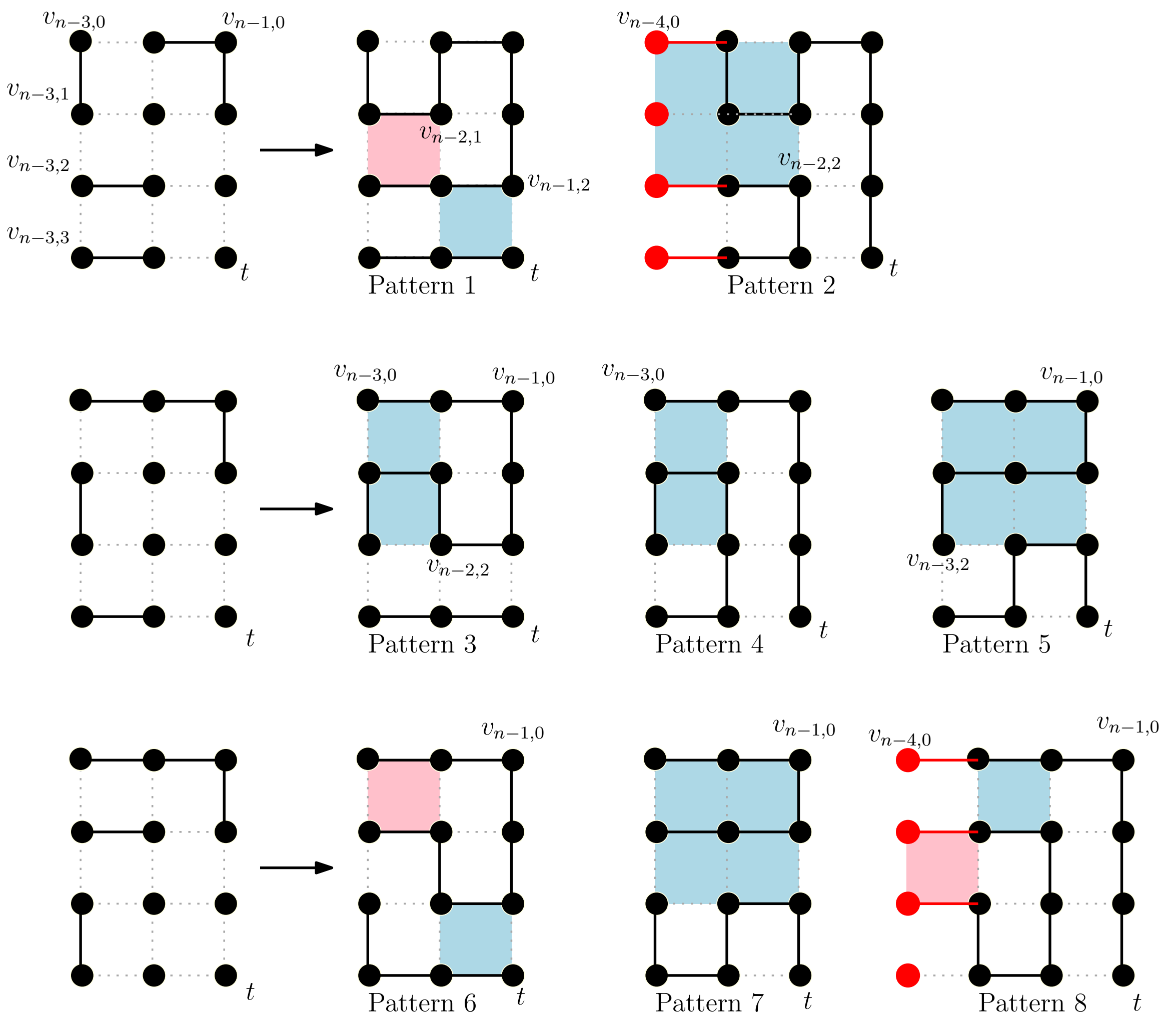


Figure 3.16: Possible configurations of the window when exactly one of the edges in Column $n-3$ is on P.

Pattern 13. P is already in the form shown in Figure 3.13; therefore, no operation is applied.

Pattern 14. We widen our window by one column to the left, then apply $Flip(v_{n-4,0}, v_{n-2,1})$.

Pattern 15. P is already in the form shown in Figure 3.13; therefore, no operation is applied.

Pattern 16. P is already in the form shown in Figure 3.13; therefore, no operation is applied.

We then move the window two columns to the left and follow the same procedure on

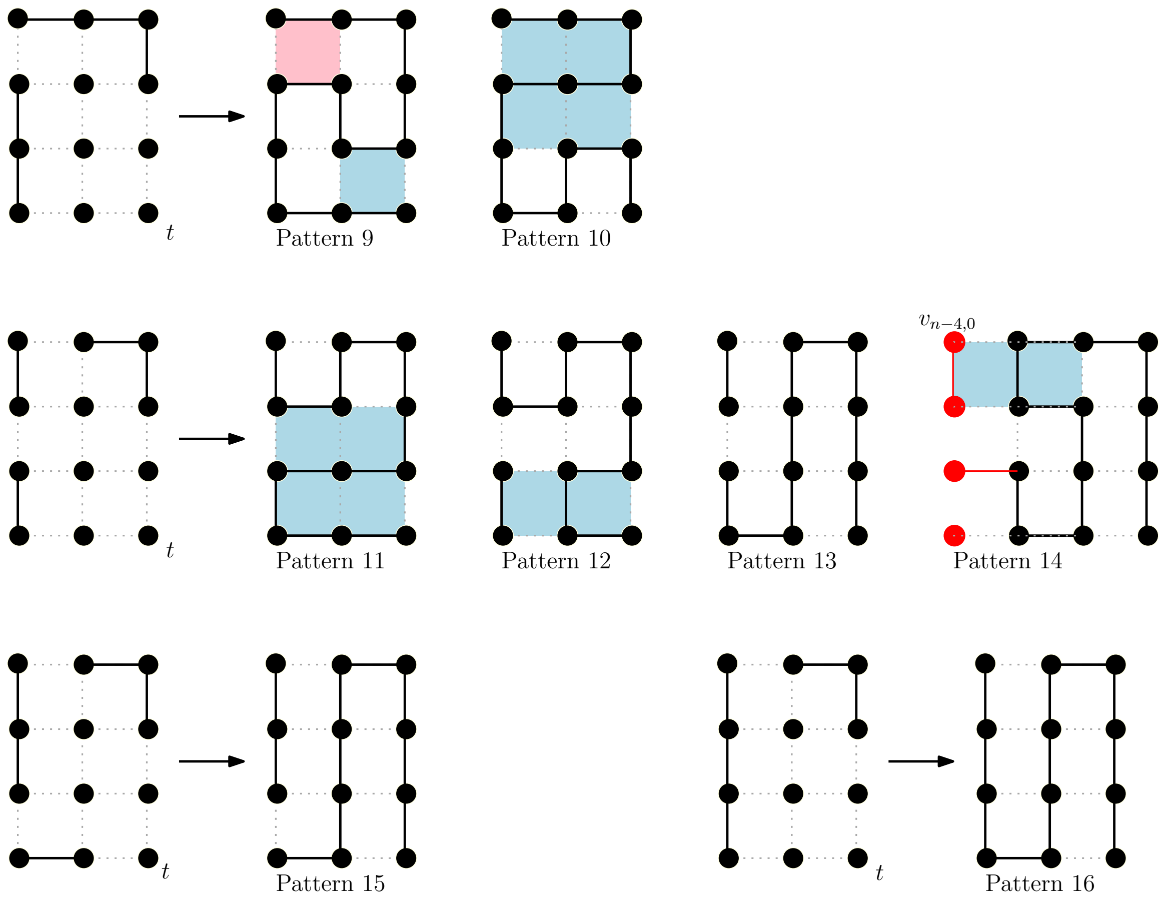


Figure 3.17: Possible configurations of the window when at least two of the edges in Column $n-3$ are on P.

Columns $n-5$, $n-4$ and $n-3$. Since $n > 3$ is odd, in the final case, the window will cover the first three columns of G, and we can only have the patterns 9, 10, 11 or 16 there. Since in each window, we apply at most 2 local operations, and there are $(n-1)/2$ windows to consider, the total number of flips, transposes and switches is at most $3(n-1)/2$.

□

Remark Figure 3.18 shows an example where the algorithm implicit in the proof of the above theorem requires exactly 1 transpose in each window to get to the canonical path $\mathbb{P}$, and hence requires $(n-1)/2$ transposes in total. Therefore, the running time of the algorithm is $\Theta(n)$.

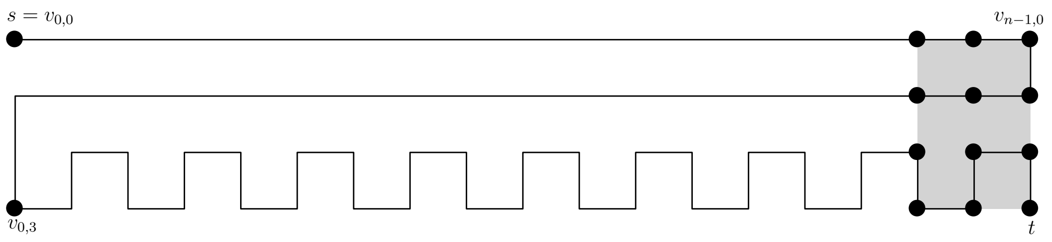


Figure 3.18: An s, t Hamiltonian path in a $4 \times n$ grid.

3.6 Conclusion

In this chapter, we proved that the cycle bend complexity of narrow grid graphs ($m \leq 4$) is constant and small ($B(G) \leq 4$), while there are s, t Hamiltonian paths in $4 \times n$ grid graphs with arbitrarily large bend complexity. We gave linear time algorithms to reconfigure Hamiltonian cycles and s, t Hamiltonian paths in narrow grid graphs, where the bend complexities of the two Hamiltonian paths or cycles are not necessarily the same. However, it seems that reconfiguration of paths is a more challenging problem than reconfiguration of cycles in general, the challenge perhaps being due to the higher bend complexity of paths than cycles in the same grid.

We introduced the concept of *cookies* in rectangular grid graphs, and we remarked that the concept and lemmas about cookies apply to more general settings such as L-shaped grids, which will be discussed in Chapter 5.

We conclude with some open problems.

1. Although we applied switch operations along with flips and transposes for reconfiguring paths, we claim that transposes and flips are sufficient for $m \leq 4$. We omit the proof, however.

2. Is the concept of cookies applicable to paths and cycles in grid graphs with holes? How can we define boundaries and cookies for grid graphs in general?

3. It would be interesting to count the number of cycles in a $4 \times n$ grid using the shapes of the cookies.

Chapter 4

1-Complex Cycles in Rectangular Grid Graphs

In this chapter we discuss Hamiltonian *cycles* with *bend complexity* 1, i.e., 1-*complex cycles*, in rectangular grid graphs. We explore the structure of such cycles for any $m, n \geq 4$, as cycles in narrow grids have been discussed in the previous chapter. In particular, we establish properties of cookies (already defined in Chapter 3) in the context of a 1-complex cycle, and define a *zip* operation, comprised of flips and transposes, that we use later in our reconfiguration algorithms. The purpose of defining the zip operation is to *grow* a certain type of cookie to a specified size while *preserving* the bend complexity of the cycle at each intermediate step. Recall from Chapter 2 that although flips and transposes preserve Hamiltonicity, they do not necessarily preserve the bend complexity.

We first give algorithms in Sections 4.2 and 4.3 to reconfigure 1-complex cycles that do not have cookies from all four boundaries of G using the *zip* operation we define in this chapter such that the bend complexity of the cycle at each intermediate step remains 1. We then give algorithms for the general case in Section 4.4, i.e., for reconfiguring 1-complex cycles with no restriction on cookies, using algorithms from Sections 4.2 and 4.3. This algorithm illustrates a powerful technique: *splitting* reconfiguration problems into smaller and solvable *subproblems*. We use the technique in the next chapter to reconfigure 1-complex cycles in L-shaped grid graphs (see Chapter 5).

The rest of the chapter is organized as follows.

1. In Section 4.1, we explore properties of cookies in 1-complex cycles. We then define

a *zip* operation that is comprised of flips and transposes.

2. In Section 4.2, we give an algorithm to reconfigure any *canonical* Hamiltonian cycle to any other *canonical* Hamiltonian cycle using zip operations. The algorithm uses $O(|G|)$ flips and transposes and preserves the bend complexity of the Hamiltonian cycle at each intermediate step.

3. In Section 4.3, we achieve results similar to Section 4.2 for 1-complex cycles with at most three types of cookies by giving a scanline algorithm. This reconfiguration algorithm also uses $O(|G|)$ flips and transposes and preserves the bend complexity of the Hamiltonian cycle at each intermediate step.

4. In Section 4.4, we give an algorithm to reconfigure *any* 1-complex Hamiltonian cycle, i.e., with at most four types of cookies, to any other 1-complex Hamiltonian cycle using $O(|G|)$ flips and transposes, building on the results from Sections 4.2 and 4.3.

5. Section 4.5 concludes the chapter by summarizing the results of this chapter and discussing some open problems.

4.1 1-Complex Cookies and Zip

Let C be a 1-complex Hamiltonian cycle in a rectangular grid graph G, and let c be a cookie of C. Recall from Chapter 3 that a cookie c of C is a sequence of consecutive vertices on C that begins and ends on adjacent non-corner vertices on a boundary of G and that does not contain any other vertices on the boundaries of G. We now define some terminology and observe some properties of c that are special to cookies in a 1-complex cycle.

Observation 4.1. *Let c be a cookie of a 1-complex cycle C of G. Then c bends exactly twice at two internal vertices of G that are adjacent in G.*

Proof. Let (p, q) be the base of c, and let u be the vertex where c bends for the first time after leaving the boundary of G at p. Let v be the vertex c visits after u. Then c must bend at v and connect straight to q, as otherwise the vertex v would have bend complexity at least 2 which is a contradiction to C being 1-complex. □

Therefore, c must outline a rectangle such that one of its dimensions is 1 as shown in Figure 4.1.

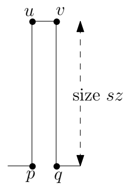


Figure 4.1: An S cookie in a 1-complex cycle.

Definition 4.1 (Size of a cookie)**.** *The* size sz *of a* 1*-complex cookie is the distance it extends along its track* tr*.*

Observation 4.2. *The vertices of* c *lie on a* $1 \times sz$ *rectangle* R*, where* sz *is the size of* c*.* R *is contained in a horizontal track* t^h_y*,* $0 < y < m - 2$*, when* c *is an E or W cookie, or in a vertical track* t^v_x*,* $0 < x < n - 2$*, when* c *is an N or S cookie.*

Definition 4.2 (Set of cookies)**.** *A* set of cookies *is a group of cookies of the same type (N,S, E or W) and of the same size that are consecutive on* C*.*

In Chapter 3, we defined a *canonical Hamiltonian cycle* in an $m \times n$ grid graph, where $m, n > 2$, as a 1-complex Hamiltonian cycle that has exactly one type of cookie. Here we reword the definition as follows: a *canonical Hamiltonian cycle* is a 1-complex Hamiltonian cycle that has exactly one *set* of cookies. Later, when we consider L-shaped grid graphs in Chapter 5, we will define *canonical cycles* in terms of *sets* of cookies, so we use the terminology here by way of introduction. For the case of rectangular grid graphs, the canonical cycles do not change. Since by Observation 4.2, a cookie in a 1-complex cycle outlines a $1 \times sz$ rectangle, where sz is the size of the cookie, the canonical cycles in rectangular grid graphs are as shown in Figure 4.2.

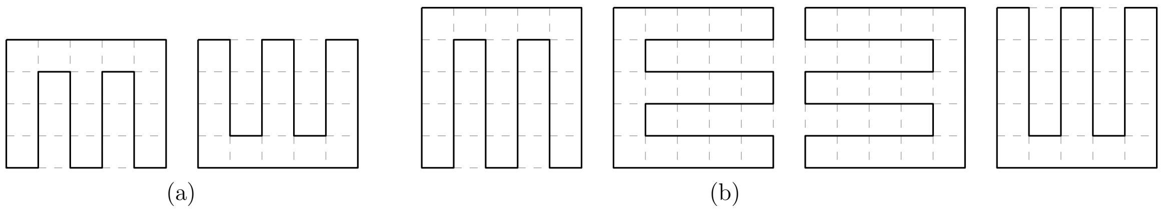


Figure 4.2: (a) Two canonical cycles on a 5×6 grid graph. (b) Four canonical cycles on a 6×6 grid graph.

We now define an operation that we call "zip", which we later apply on 1-complex Hamiltonian cycles to *grow* cookies of certain types.

Definition 4.3 (Zip). *A* zip *operation $Z = zip_{tp}(tr, sz)$ in C of G is a sequence of zero or more transpose operations followed by zero or more flip operations in track tr such that the size of the cookie of type tp in track tr is sz after the zip, where sz does not exceed that maximum size that a cookie of type tp can have, i.e., $n - 2$ for E and W cookies, and $m - 2$ for N and S cookies.*

To ensure that a zip preserves the bend complexity of the cycle after each operation in the zip, our reconfiguration algorithms only perform zip under certain conditions. To state these conditions, we define the *zone* of a zip operation.

Definition 4.4 (Zone of a Zip). *The* zone *of a zip $Z = zip_{tp}(tr, sz)$ is the closed $1 \times sz$ rectangle in track tr that contains the desired cookie of type tp and size sz (see Figure 4.3).*

For the next definition, consider the two closed line segments s and s' that are $\frac{1}{2}$-unit translates of the sz-length sides of the zone, such that s and s' are outside the zone and each has one endpoint on the tp boundary. We call them the two *sidelines* of the zone. Now we can define *zippability*, which gives the conditions under which we will perform zips.

Definition 4.5 (Zippability). *The zone of $Z = zip_{tp}(tr, sz)$ is* zippable *if*

(i) at least one of its sidelines does not intersect any cookie of C,

(ii) any cookie perpendicular to tr that intersects the zone covers exactly 4 internal vertices of G in the zone, and

(iii) any cookie of C of type tp that lies in tr at the start of a possible zip has size $\leq sz$, the desired cookie size, after the zip.

Figures 4.3(a)–(b) show examples of zippable zones and Figures 4.3(c)–(d) show examples of zones that are not zippable. The zone in Figure 4.3(c) is not zippable as it violates Property (ii) of Definition 4.5; both its sidelines intersect cookies. The zone in Figure 4.3(d) is not zippable as it violates Property (iii) of Definition 4.5; a cookie perpendicular to track tr intersects only two vertices in the zone, not four.

The following lemma justifies the definition of zippability for 1-complex Hamiltonian cycles.

Lemma 4.1. *Let C be a 1-complex Hamiltonian cycle in a rectangular grid graph G and let $Z = zip_{tp}(tr, sz)$ be a zip operation on C. If the zone of Z is zippable, then*

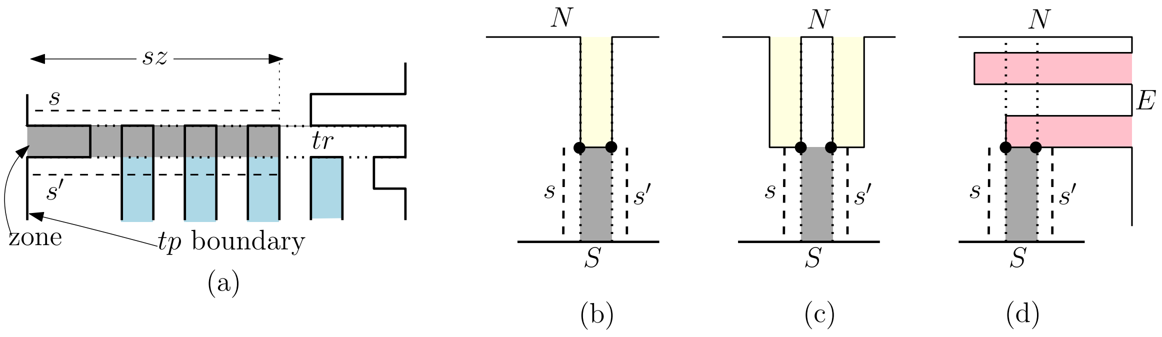


Figure 4.3: Example of zippable and unzippable zones, where zones are shown in gray, and cookies are colored as follows: yellow for N cookies, pink for E cookies and blue for S cookies. The zones in (a)–(b) are zippable, but the zones in (c)–(d) are not zippable. The complete cycle C is not shown. Sidelines are shown dashed.

(a) Z reconfigures C to another 1-complex Hamiltonian cycle C' such that C' has a cookie of type tp and size sz in tr,

(b) the cycle remains 1-complex after each transpose and flip operation of Z, and

(c) in total, at most sz flips and transposes are performed in Z.

Proof. Before proving the claims we establish the structure of the zippable zone of Z.

Since the zone of Z is zippable, it must have the following structure. If there is a cookie c of type tp in track tr then c must have size $\leq sz$ by Property (iii) of Definition 4.5. The cookie c then covers all or some of the vertices of the zone closest to the boundary tp. Otherwise, if there is no cookie of type tp in tr, then there is an edge on boundary tp in tr. We can consider this case as tr having the cookie c, where the size of c is 0. If the zone has a cookie c' from the boundary opposite to tp, c' must cover some (or all) the vertices of tr farthest from the boundary tp. If c and c' together do not cover all the internal vertices of the zone, then there must be a number I of cookies perpendicular to tr covering those remaining vertices of the zone. Since the zone is zippable and one of the sidelines of the zone does not intersect any cookies (Property (i) of Definition 4.5), the perpendicular cookies must all be based on the same boundary (east, west, north or south) and must each cover exactly 4 vertices of the zone (Property (ii) of Definition 4.5). See Figure 4.3(a).

We now prove claims (a)–(c). Without loss of generality, assume that the track tr is a horizontal track t^h_y, and that the sideline s that does not intersect any cookie that lies just above Row y, and that the cookie c is a W cookie of size x, where $0 \leq x \leq sz$, as shown in Figure 4.3(a).

(a) We prove this claim by induction on the number I of cookies that lie in tracks perpendicular to tr and that intersect the zone of the zip Z.

If $I = 0$, then tr is covered by just c, or just by c', or by c and c' together. Then we apply $Flip(v_{x,y}, v_{x+2,y+1})$ and set $x = x+1$. The flip operation increases the size of c by 1 and decreases the size of c' by 1. We repeat this step until $x = sz$, completing the base case of the induction.

Suppose that Claim (a) holds for $I \leq k$, where $k \geq 0$. We now assume that $I = k + 1$. Then there must be a perpendicular cookie c'' in vertical track t^v_{x+1} that intersects the track tr and covers exactly 4 internal vertices of G in the zone. According to our assumption about track tr and cookie c, the cookie c'' must be an S cookie. We can apply $Xpose(v_{x,y+2}, v_{x+2,y})$, reducing the size of c'' by 2 and increasing the size of c by 2. The cycle obtained after this transpose has k perpendicular cookies intersecting the zone. Hence, we are done by induction.

(b) We prove this claim by demonstrating that each transpose and flip operation in Z produces a 1-complex cycle. Recall from Chapter 2 that flip and transpose preserve Hamiltonicity. Here we prove that, because the zippability condition holds, the Hamiltonian cycle obtained after a single flip or a single transpose of Z is 1-complex. We consider two cases from the proof of Claim (a), $I = 0$ and $I > 0$.

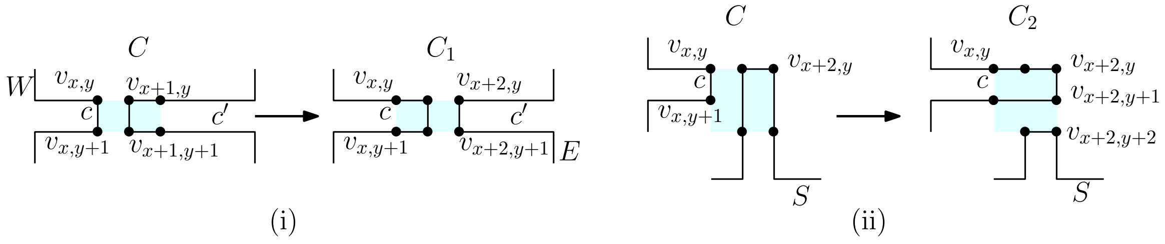


Figure 4.4: The case when (i) $I = 0$ and we apply a flip in Z, and (ii) $I > 0$ and we apply a transpose in Z. The blue highlighting shows the 2×3 and 3×3 subgrids for the operations; for transpose operations, the 3×3 subgrid does not lie completely inside the track tr where the S cookie is expanding.

When $I = 0$, either the size of the W cookie c is $x = 0$ and it has no bends, or the size of c is $x > 0$ and by Observation 4.1, it bends at exactly two vertices $v_{x,y}$ and $v_{x,y+1}$; see Figure 4.4(i). If there is an E cookie c' in $tr = t^h_y$, then it bends at exactly at two vertices $v_{x+1,y}$ and $v_{x+1,y+1}$. We perform $Flip(v_{x,y}, v_{x+2,y+1})$ to obtain a Hamiltonian cycle C_1. In C_1 the W cookie c now bends at exactly two vertices $v_{x+1,y}$ and $v_{x+1,y+1}$. If the size of E cookie c' is not zero in C', then it must bend at exactly two vertices $v_{x+2,y}$ and $v_{x+2,y+1}$. Therefore, all the internal vertices on c and c' are still connected to a boundary of G by a bend-free subpath on C_1 as is every other internal vertex. Therefore, C_1 is a 1-complex Hamiltonian cycle. In this way we can prove that the cycle obtained after each such flip operation of Z is 1-complex.

When $I > 0$, the first operation we perform is a transpose operation $Xpose(v_{x,y+2}, v_{x+2,y})$. Let the cycle obtained after this operation be C_2. In C, either the size of the W cookie c is $x = 0$ and it has no bends, or the size of c is $x > 0$ and by Observation 4.1, it bends at exactly two vertices $v_{x,y}$ and $v_{x,y+1}$. The S cookie in track t^v_{x+1} bends at exactly two vertices $v_{x+1,y}$ and $v_{x+2,y}$; see Figure 4.4(ii). In C_2, c bends at exactly two vertices $v_{x+2,y}$ and $v_{x+2,y+1}$. There is either no S cookie in track t^v_{x+1} in C_2, or an S cookie that bends at exactly two vertices $v_{x+1,y+2}$ and $v_{x+2,y+2}$. Since every internal vertex on these cookies is connected to a boundary of G by a bend-free subpath on C_2, cycle C_2 is a 1-complex Hamiltonian cycle. It is straightforward to prove that the cycle remains 1-complex after each remaining transpose of Z.

(c) Each transpose and flip increases the size of c by 2 and 1, respectively. Since the size of c is sz after Z is completed, the number of flip and transpose operations is at most sz. □

4.2 Reconfiguration between Canonical Cycles

In this section we show, by giving an algorithm, that any canonical Hamiltonian cycle $\mathbb{C}_1$ of an $m \times n$ grid graph G can be reconfigured to any other canonical Hamiltonian cycle $\mathbb{C}_2$ of G.

Theorem 4.1. *Given two canonical Hamiltonian cycles $\mathbb{C}_1$ and $\mathbb{C}_2$ on an $m \times n$ grid graph G, $\mathbb{C}_1$ can be reconfigured to $\mathbb{C}_2$ by applying at most $O(|G|)$ flip and transpose operations.*

Proof. First we provide an algorithm for carrying out the reconfiguration and then we count the number of flip and transpose operations it performs.

Without loss of generality, assume that $\mathbb{C}_1$ has W cookies. We consider the following two cases.

Case 1: $\mathbb{C}_2$ has E cookies. In this case m must be even. We apply a zip $zip_E(t^h_1, n-2)$ in track t^h_1 of $\mathbb{C}_1$ to obtain an E cookie of size $n-2$. We then move to track t_3 and apply zip $zip_E(t_3^h, n-2)$. We continue this process until there are no more W cookies. See Figure 4.5.

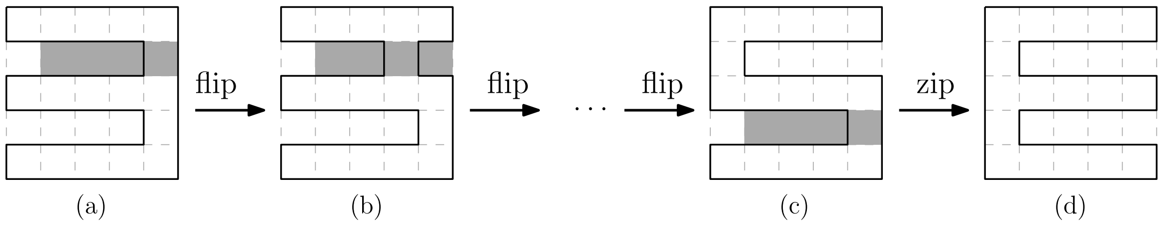


Figure 4.5: Reconfiguring a canonical Hamiltonian cycle with W cookies to another canonical Hamiltonian cycle with E cookies.

Case 2: $\mathbb{C}_2$ has N cookies. In this case both m and n must be even. We apply a zip $zip_N(t_{n-3}^v, m-2)$ in the vertical track t_{n-3}^v to obtain an N cookie of size $m-2$ in that track. We then move to track t_{n-5} and apply a similar zip, $zip_N(t_{n-5}^v, m-2)$. We continue this process until we have only N cookies. See Figure 4.6.

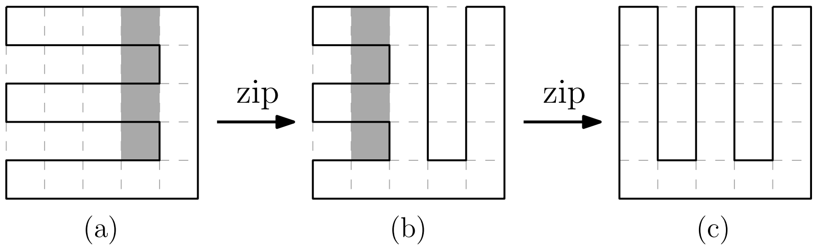


Figure 4.6: Reconfiguring a canonical Hamiltonian cycle with W cookies to another canonical Hamiltonian cycle with N cookies.

The case when $\mathbb{C}_2$ has S cookies is similar to Case 2, and if $\mathbb{C}$ has W cookies there is nothing to do. In all the cases above, we need at most $O(|G|)$ flips and transposes by Lemma 4.1. □

We refer to the algorithm in the proof of Theorem 4.1 as Algorithm CanToCan, so called because we are reconfiguring a *canonical* cycle to another *canonical* cycle.

4.3 Reconfiguration of Cycles with Three Types of Cookies

Let G be an $m \times n$ rectangular grid graph and let C_1 and C_2 be two 1-complex Hamiltonian cycles of G with at most three types of cookies each. In this section, we give an algorithm to reconfigure C_1 to C_2 using a linear number of flips and transposes.

We first give an overview of our approach, which is to use canonical forms as intermediate configurations. Let $\mathbb{C}_1$ and $\mathbb{C}_2$ be two distinct canonical Hamiltonian cycles of G. We show that C_1 can be reconfigured to C_2 using only flip and transpose operations in four steps as shown in Figure 4.7:

- **Step (a)** we first reconfigure C_1 to $\mathbb{C}_1$,
- **Step (b)** we reconfigure C_2 to $\mathbb{C}_2$,
- **Step (c)** if $\mathbb{C}_1 \neq \mathbb{C}_2$, we then reconfigure $\mathbb{C}_1$ to $\mathbb{C}_2$, and
- **Step (d)** finally, we reverse the steps of (b) to reconfigure $\mathbb{C}_2$ to C_2.

Effectively, we reconfigure C_1 to $\mathbb{C}_1$ to $\mathbb{C}_2$ to C_2 as shown in Figure 4.7. It suffices to give algorithms for Steps (a) and (c) only.

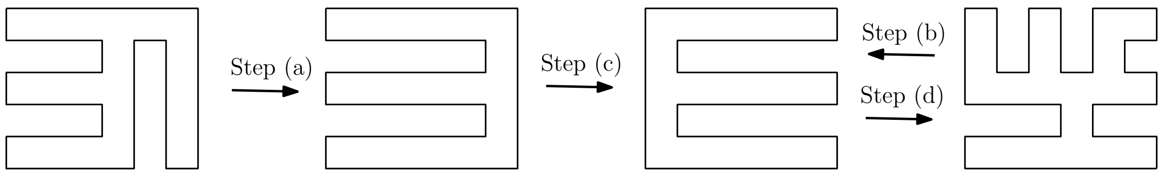


Figure 4.7: Steps (a) and (b): reconfiguring 1-complex Hamiltonian cycles C_1 and C_2 to canonical Hamiltonian cycles $\mathbb{C}_1$ and $\mathbb{C}_2$, respectively. Step (c): reconfiguring $\mathbb{C}_1$ to $\mathbb{C}_2$. Step (d): reversing Step (b).

In Section 4.2, we gave an algorithm for transforming any canonical Hamiltonian cycle to another canonical Hamiltonian cycle (Step (c)). We now give an algorithm, which we call Algorithm RectThreeTypes, to reconfigure C_1 to a canonical Hamiltonian cycle (Step (a)). We then show that C_1 can be reconfigured to C_2 using a linear number of flips and transposes in Theorem 4.3.

We begin with an overview of Algorithm RectThreeTypes. Let C be a 1-complex Hamiltonian cycle on an $m \times n$ grid graph G such that C has at most three types of cookies. Since we can rotate G by multiples of 90° and possibly swap the values of m and n, we can assume that C has no E cookies. Thus all the edges on the E boundary, which is also column $n-1$, must be on C. The only bends are at the corner vertices $v_{n-1,0}$ and $v_{n-1,m-1}$.

We start from track t^v_{n-3} and scan the vertices from top (y-coordinate 0) to bottom (y-coordinate $m-1$). Scanning down the track, we note that the internal vertices must be covered by an N cookie until either the N cookie ends one row above the south boundary, or until the first encounter with a W or S cookie. If a W cookie is met before any S cookie, subsequent internal vertices must be covered by W cookies, each of which covers exactly four vertices of t^v_{n-3}, until either an S cookie is met or the south boundary is reached. If an S cookie is met, it contains all remaining internal vertices of t^v_{n-3}. Then the zone of $Z = zip_N(t^v_{n-3}, m-2)$ is zippable by Definition 4.5 since the sideline between Column $n-2$ and the E boundary does not intersect any cookies satisfying Property (i), any W cookie intersecting t^v_{n-3} covers exactly 4 vertices satisfying Property (ii), and any N cookie in this track must have size $\leq m-2$, which is the maximum size for any N cookie in C, satisfying Property (iii).

We apply the zip Z as demonstrated by an example in Figure 4.8. We then move two tracks to the left to track t^v_{n-5}. We can prove in a way similar to the above that the zone of zip $zip_N(t^v_{n-5}, m-2)$ is zippable as the sideline between Columns $n-4$ and $n-3$ does not intersect any cookies. We apply the zip and move two tracks to the left. We repeat the process until we reach Column 1 or Column 0 (the W boundary). Observe that at any time, the internal vertices in the vertical tracks that have already been processed are covered by N cookies.

If n is even, we must reach Column 0 eventually, since we start at track t^v_{n-3} and move two tracks to the left at each step. Then the algorithm ends as the Hamiltonian cycle now has only N cookies, which means that we have obtained a canonical Hamiltonian cycle.

Otherwise, we reach Column 1. By Lemma 4.1, we still have a 1-complex Hamiltonian cycle as we performed each zip in a zippable zone. Therefore, in this case the internal vertices in Column 1 must be covered by only W cookies, since an N or S cookie requires two columns. We then apply zip operations with tp =W. We start with the bottommost horizontal track t^h_{m-3}. The zone of $zip_W(t^h_{m-3}, n-2)$ is zippable by Definition 4.5 since the sideline between Row $m-2$ and the S boundary does not intersect any cookies (Property (i)), no S or E cookies intersect the track and any N

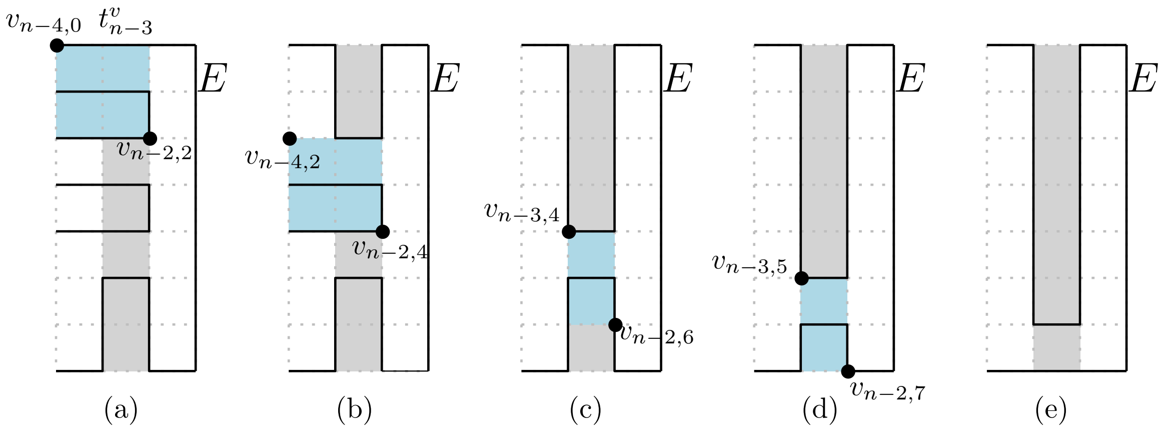


Figure 4.8: Steps of zip operation $Z = zip_N(t^v_{n-3}, m-2)$: (a) apply $Xpose(v_{n-4,0}, v_{n-2,2})$, (b) apply another transpose $Xpose(v_{n-4,2}, v_{n-2,4})$, (c) then $Flip(v_{n-3,4}, v_{n-2,6})$, (d) apply another flip $Flip(v_{n-3,5}, v_{n-2,7})$, and (e) arrive at the final state.

cookie intersecting t^h_{m-3} covers exactly 4 vertices satisfying Property (ii), and the W cookie in t^h_{m-3} has size $1 \leq n-2$, which is the maximum size for any W cookie in C, satisfying Property (iii). We apply the zip and move two tracks above to track t^h_{m-5}. The zone of $zip_W(t^h_{m-5}, n-2)$ is now zippable by a similar logic as above. Therefore, we apply the zip. We repeat the process until we reach Row 0, which must be the only case as m must be even in the case of n being odd. We now have a Hamiltonian cycle with only W cookies, which is a canonical Hamiltonian cycle.

We give the pseudocode for this algorithm in Algorithm RectThreeTypes and then prove its correctness in Theorem 4.2.

Algorithm *RectThreeTypes*(G, C)

Input: An $m \times n$ grid graph G, $m, n \geq 3$ where at least one of m, n is even, and a 1-complex Hamiltonian cycle C of G with no E cookies.

Output: A canonical Hamiltonian cycle of G with only one set of cookies of type N or W.

```
1.  Set $x = n - 3$;              //$t^v_{n-3}$ is the easternmost vertical track
2.                                //that can contain an N or S cookie.
3.  while $x > 1$
4.          $C = zip_N(t^v_x, m - 2)$;    //grow an N cookie of maximum size
5.          $x = x - 2$;            //move two tracks to the left, and repeat
6.  if $x = 1$
7.      then Set $y = m - 3$;     //$n$ must be odd and
8.                                //Column 1 is covered by unit size W cookies.
9.  while $y > 0$
10.         $C = zip_W(t^h_y, n - 2)$;    //grow a W cookie of maximum size
11.         $y = y - 2$;            //move two tracks above, and repeat
12. return $C$
```

Theorem 4.2. *Let C be a 1-complex Hamiltonian cycle on an $m \times n$ grid graph G with no E cookies. Then Algorithm*RectThreeTypes *reconfigures C to a canonical Hamiltonian cycle $\mathbb{C}$ of G using $O(|G|)$ flips and transposes, where $\mathbb{C}$ has only one type of cookie, N or W.*

Proof. We first assume that n is even . Then we apply zips only along the vertical tracks $t^v_{n-3}, t^v_{n-5}, \ldots, t^v_1$. The zone of $zip_N(t^v_{n-3}, m-2)$ is zippable by Definition 4.5 since the sideline between Column $n-2$ and the E boundary does not intersect any cookies satisfying Property (i), any W cookie intersecting t^v_{n-3} covers exactly 4 vertices satisfying Property (ii), and any N cookie in this track must have size $\leq m-2$, which is the maximum size for any N cookie in C, satisfying Property (iii). After applying $zip_N(t^v_{n-3}, m-2)$, we obtain a 1-complex Hamiltonian cycle by Lemma 4.1, and the zone of the next zip $zip_N(t^v_{n-5}, m-2)$ is zippable in the newly obtained 1-complex cycle. Therefore, we apply the zip in t^v_{n-5} and move to t^v_{n-7}. In this way every zip gives a 1-complex Hamiltonian cycle where the zone of the next zip is zippable. Eventually we reach the W boundary, and all the internal vertices in the columns between the E and W boundaries are covered by N cookies only.

We now assume that n is odd. Then m must be even as otherwise there won't be a Hamiltonian cycle of G. We apply zips along the vertical tracks $t^v_{n-3}, t^v_{n-5}, \ldots, t^v_2$ as above and obtain a 1-complex Hamiltonian cycle C' by Lemma 4.1. C' has N cookies of size $m-2$ covering the internal vertices of tracks $t^v_{n-3}, t^v_{n-5}, \ldots, t^v_2$. Since any N or S cookie needs two columns, Column 1 of C' must be covered by only W cookies and they must have unit size each. We then apply zips along the horizontal tracks $t^h_{m-3}, t^h_{m-5}, \ldots, t^h_1$. The zone of $zip_W(t^h_{m-3}, n-2)$ is zippable by Definition 4.5 since the sideline between Row $m-2$ and the S boundary does not intersect any cookies satisfying Property (i), any N cookie intersecting t^h_{m-3} covers exactly 4 vertices satisfying Property (ii), and the W cookie in t^h_{m-3} has size $1 \leq n-2$, which is the maximum size for any W cookie in C, satisfying Property (iii). We apply the zip to obtain another 1-complex Hamiltonian cycle (by Lemma 4.1) where the zone of $zip_W(t^h_{m-5}, n-2)$ is zippable. In this way we get a 1-complex Hamiltonian cycle after every zip where the zone of the next zip is zippable. We finally get a Hamiltonian cycle with only W cookies of size $n-2$.

We apply $\lfloor n/2 \rfloor$ zips along vertical tracks and each such zip needs at most $m-2$ flips and transposes by Lemma 4.1. Similarly the zips along the horizontal tracks (when n is odd) need at most $n-2$ flips and transposes each by Lemma 4.1. Therefore, the total number of local operations in Algorithm RectThreeTypes is $O(|G|)$. $\square$

We now summarize our result in the following theorem.

Theorem 4.3. *Any* 1*-complex Hamiltonian cycle* C_1 *of* G *with at most three types of cookies can be reconfigured to any other* 1*-complex Hamiltonian cycle* C_2 *with at most three types of cookies in* $O(|G|)$ *time using* $O(|G|)$ *flip and transpose operations.*

Proof. The given 1-complex Hamiltonian cycles C_1 and C_2 can each be reconfigured to canonical Hamiltonian cycles $\mathbb{C}_1$ and $\mathbb{C}_2$, respectively, using $O(|G|)$ flips and transposes by Theorem 4.2. $\mathbb{C}_1$ can be reconfigured to $\mathbb{C}_2$, when $\mathbb{C}_1 \neq \mathbb{C}_2$, using $O(|G|)$ flips and transposes by Theorem 4.1. Therefore, the total number of flips and transposes required to reconfigure C_1 to C_2 is $O(|G|)$. Since each local operation takes $O(1)$ time, it takes $O(|G|)$ time to reconfigure C_1 to C_2. $\square$

4.4 Reconfiguration of Cycles with Four Types of Cookies

In this section, we show how to create certain subproblems we can solve with the algorithms we already have, and how to merge their solutions to obtain reconfiguration of Hamiltonian cycle with all four types of cookies.

We first give a formal definition of a "problem" which will lead to "subproblems".

Definition 4.6 (Problem $\Pi_R(C, G)$)**.** *Given a 1-complex Hamiltonian cycle C in a rectangular grid graph G, reconfigure C to a canonical Hamiltonian cycle $\mathbb{C}$ of G.*

The subscript on Π indicates that the shape of G is rectangular. When L-shaped grids are considered in Chapter 5, we will introduce an L subscript to indicate that the grid graph is L-shaped.

In this section we give an algorithm that we call RectFourTypes to solve $\Pi_R(C, G)$. The algorithm runs in three steps.

1. First it creates "subproblems of $\Pi_R(C)$", which we define in Section 4.4.1. The key idea is to define *subproblems* that can be handled by the algorithms we already have, which reconfigure cycles having at most three types of cookies.

2. Then it solves the *subproblems* using the algorithms from Section 4.3.

3. Finally, it merges the solutions to the *subproblems* to obtain $\mathbb{C}$.

4.4.1 Creating Subproblems

We first define some terminology that we use to define a *subproblem of* $\Pi_R(C, G)$.

Definition 4.7 (Splitting Track)**.** *A track t of G is a* splitting track *of C if the interior of t does not intersect any cookies of C and C has cookies on each side of t (i.e., both above and below t, or both on the left and on the right of t).*

Let $t = t_i^h$ be a splitting track of C as shown in Figure 4.9(a). We remove the two edges (u, u') and (v, v') of C at the two ends of t on the W and E boundaries of G, to partition C into two disjoint paths P_{uv} and $P_{u'v'}$. Let the rectangular subgrids (defined in Chapter 2) of G that contain P' and P'' be G' and G'', where G' is above t and G'' is below. Now add a copy of Row i from u to v (inclusive) one unit below G' to create the augmented vertex-induced grid graph $(G')^+$. Join u to v by a path through the new vertices to obtain a 1-complex Hamiltonian cycle C' of $(G')^+$ (see Figure 4.9(b)). Similarly obtain a Hamiltonian cycle C'' of $(G'')^+$ (see Figure 4.9(c)), the vertex-induced grid graph created from G'' by adding a row above. The rows added to create $(G')^+$ and $(G'')^+$ are their *augmented boundaries.*

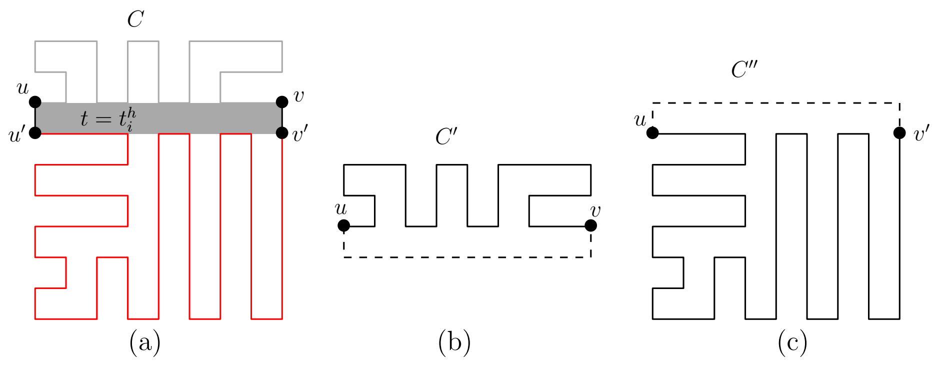


Figure 4.9: (a) 1-complex Hamiltonian cycle C, where splitting track $t = t_i^h$ is shown in gray. (b) 1-complex Hamiltonian cycle C' of $(G')^+$, and (c) 1-complex Hamiltonian cycle C'' of $(G'')^+$.

Under the assumption that some splitting track has been found, we now formally define *subproblems* of $\Pi_R(C, G)$.

Definition 4.8 (Subproblems). $\Pi_R(C', (G')^+)$ *and* $\Pi_R(C'', (G'')^+)$ *are the* subproblems *of* $\Pi_R(C, G)$.

We observe some properties of C in the following lemma, which we use in the lemma after it to show that there always exists a splitting track if C contains cookies from all four boundaries.

Lemma 4.2. *If there exists a Column x that intersects an E and a W cookie, then there must exist a pair of E and W cookies intersecting Column x such that if the E cookie is on track t_y^h, then the west cookie must be on track t_{y+2}^h or t_{y-2}^h.*

Proof. Let the closest pair of W and E cookies intersecting Column x be in tracks t_y^h and $t_{y'}^h$, respectively. Without loss of generality, we assume that $y' > y$ (see Figure 4.10). Suppose for a contradiction that $y' > y + 2$. Then there would be some vertices of G on Column x in Rows $y + 2, \ldots, y' - 1$ that are not covered by any cookies since no N or S cookies can cover them, and there are no E or W cookies between tracks t_y^h and $t_{y'}^h$, a contradiction. Therefore, $y' = y + 2$. □

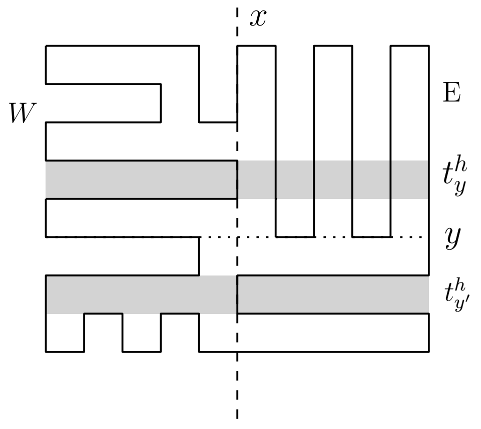


Figure 4.10: Illustration for the proof of Lemma 4.2.

Lemma 4.3. (Splitting track) *Let C be a 1-complex Hamiltonian cycle in a rectangular grid graph G, where C has cookies from all four boundaries: E, W, N and S. Then there is a splitting track t of C.*

Proof. We sweep a vertical line i, column by column, from Column 1 to Column $n-2$ until one of the following two events occurs:

1. **Sweepline i intersects both E and W cookies.** By Lemma 4.2 there must be a pair of E and W cookies such that if one of them is in track t^h_j then the other is in t^h_{j+2}, where $1 \le j \le m-4$. Then the track t^h_{j+1} does not intersect any cookies, and C has cookies both above and below t^h_{j+1}. Therefore, t^h_{j+1} is a splitting track t. See Figure 4.11(a).

2. **Sweepline i does not intersect any W cookies.** There must be at least one W cookie in C since it has cookies from all four boundaries. Then Column $i-1$ is intersected by the longest W cookie since Column i is the first column that does not intersect any W cookies. Column $i-1$ does not intersect any E cookies or we would have been in Case 1.

   If track t^v_{i-1} does not contain any cookies, then it is a splitting track. Note that $i < n-1$ since C has at least one E cookie by our assumption. Therefore, t^v_{i-1} must have a W cookie on the left and an E cookie on its right. See Figure 4.11(b).

   Otherwise, t^v_{i-1} contains at least one N or S cookie. Without loss of generality, we assume that t^v_{i-1} contains an N cookie. Let the size of the N cookie be y. Then $y < m-3$ since Column $i-1$ is intersected by at least one W cookie. If there are

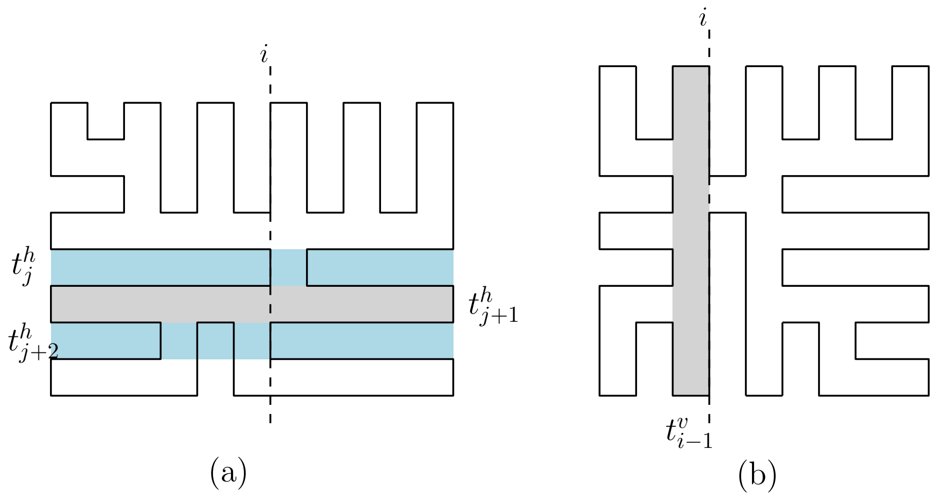


Figure 4.11: (a) Sweepline i intersecting both E and W cookies. (b) i does not intersect any W cookies and t^h_{i-1} does not contain any N or S cookies.

no S cookies intersecting Column $i-1$ then the vertices in Column $i-1$ from Row $y+1$ to $m-2$ must be covered by W cookies as no E cookie intersects Column $i-1$. If there is an S cookie intersecting Column $i-1$, then the vertices in Column $i-1$ between the N and S cookies must be covered by W cookies. In either case the vertex $v_{i-1,y+1}$ is covered by a W cookie lying in the horizontal track t^h_{y+1}. Depending on the base of the cookie covering the vertex $v = v_{i,y+1}$ in Column i just below the N cookie, we have to consider the following two cases.

(a) *Vertex v is covered by an E cookie.* Then that cookie must also be in track t^h_{y+1} and the horizontal track t^h_y is a splitting track. See Figure 4.12(a).

(b) *Vertex v is covered by an S cookie.* That S cookie must be in track t^v_i. We keep moving i to the right until it intersects an E cookie, which must be in either track t^h_{y+1} (see Figure 4.12(b)) or in track t^h_{y-1} (see Figure 4.12(c)). In either case, the track t^h_y is a splitting track.

□

We now give an algorithm FindSplitTrack that returns the leftmost vertical splitting track (if it exists) or the southernmost horizontal splitting track of C.

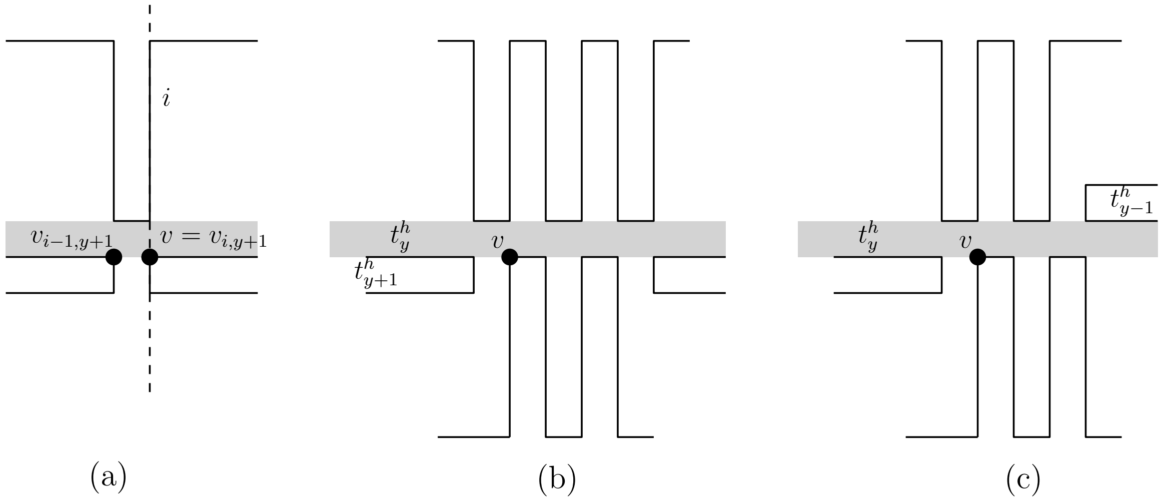


Figure 4.12: (a) The vertex $v = v_{i,y+1}$ is covered by an E cookie, (b) v is covered by an S cookie and the first E cookie encountered to the right of Column i is in track t^h_{y+1}, and (c) the E cookie is in track t^h_{y-1}.

Algorithm *FindSplitTrack*(G, C)
Input: An $m \times n$ grid graph G, and a 1-complex Hamiltonian cycle C of G
Output: A splitting track t of C

```
1. for every vertical track $t^v_x$ from $x = 2$ up to $x = n - 4$ inclusive
2.         if $t^v_x$ has cookies on both sides
3.            then return $t = t^v_x$
4.            else repeat with $x = x + 1$
5. for every horizontal track $t^h_y$ from $y = m - 4$ down to $y = 2$
6.         if $t^h_y$ has cookies on both sides
7.            then return $t = t^h_y$
8.            else repeat with $y = y - 1$
```

We prove the correctness of Algorithm FindSplitTrack in the following lemma.

Lemma 4.4. *Let C be a 1-complex Hamiltonian cycle in an $m \times n$ grid graph G such that C has cookies from all four boundaries: E, W, N and S. Then Algorithm* FindSplitTrack *finds in $O(|G|)$ time the leftmost vertical splitting track if it exists, and the southernmost horizontal splitting track otherwise.*

Proof. By Lemma 4.3, a splitting track of C exists. Therefore, if Algorithm Find-

SplitTrack does not find such a splitting track sweeping eastward, it must find a splitting track when sweeping upward. Let t be the splitting track returned by the algorithm. It is straightforward that t is either the leftmost vertical splitting track or the southernmost horizontal splitting track. Since there are two sweeps, the time required to find t is $O(|G|)$. □

4.4.2 Solving the Subproblems

After creating the subproblems $\Pi_R(C', (G')^+)$ and $\Pi_R(C'', (G'')^+)$ of $\Pi_R(C, G)$, we solve the two subproblems separately. Since both C' and C'' have no cookies from their augmented boundaries, each of them has at most three types of cookies. We position C' such that the augmented boundary is the E boundary and apply Algorithm RectThreeTypes from Section 4.3 to obtain canonical Hamiltonian cycle $\mathbb{C}'$ that has no E cookies. Similarly we obtain canonical Hamiltonian cycle $\mathbb{C}''$ from C''.

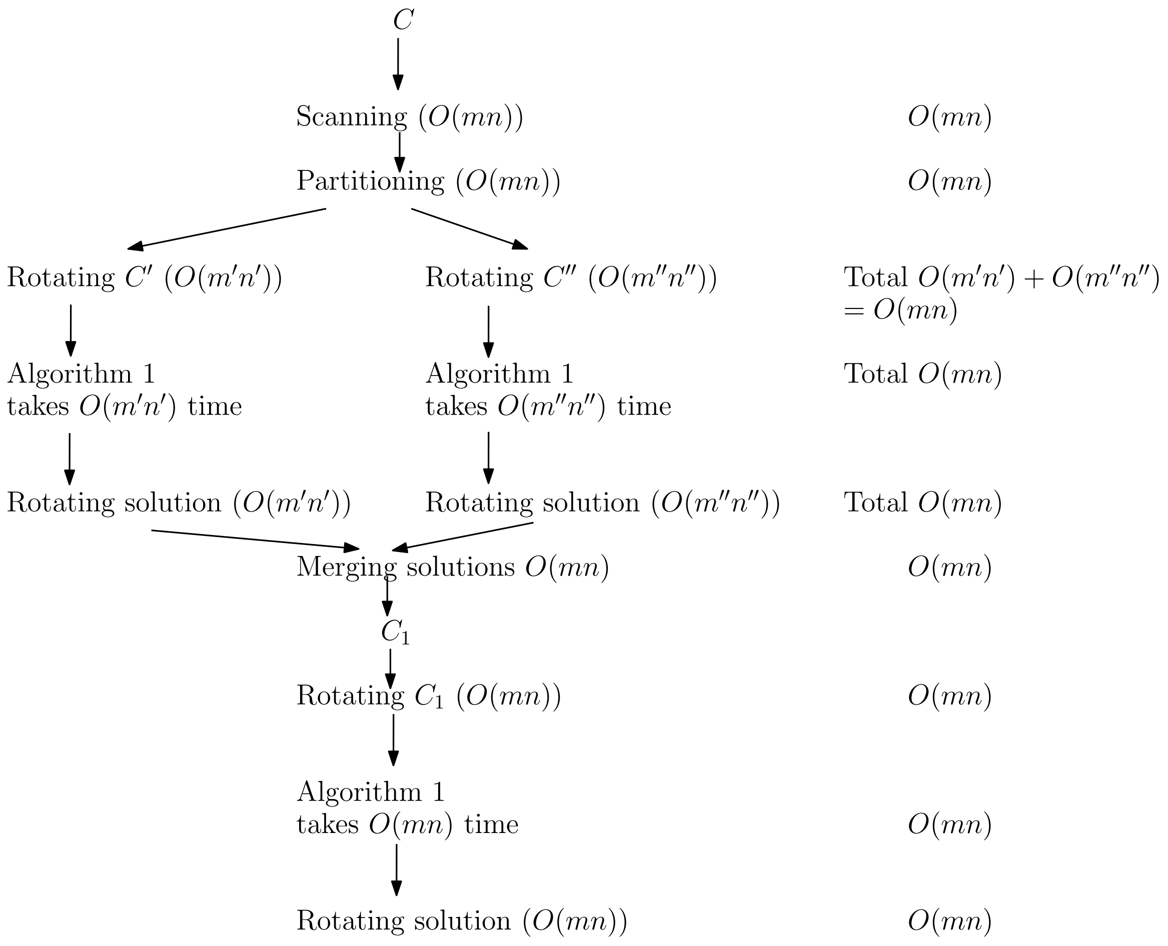
4.4.3 Merging the Solutions to the Subproblems

We merge the solutions $\mathbb{C}'$ of $\Pi_R(C', (G')^+)$ and $\mathbb{C}''$ of $\Pi_R(C'', (G'')^+)$ that are output by the algorithms as follows. We remove the augmented boundaries from $\mathbb{C}'$ and $\mathbb{C}''$ to get two Hamiltonian paths in the original subgrids of G on both sides of the splitting track. We join the paths using the boundary edges at the ends of the splitting track to obtain a 1-complex Hamiltonian cycle C_1 of G. C_1 has at most two sets of cookies. Thus C_1 has at most two *types* of cookies. We then apply the Algorithm RectThreeTypes on C_1 to obtain a canonical Hamiltonian cycle $\mathbb{C}$ of G.

Theorem 4.4. *Let C be a 1-complex Hamiltonian cycle of an $m \times n$ grid graph G. Then Algorithm* RectFourTypes *solves $\Pi_R(C, G)$ in $O(|G|)$ time using $O(|G|)$ flips and transposes.*

Proof. Figure 4.13 shows the steps of Algorithm RectFourTypes. It takes $O(|G|)$ time to create the two subproblems of $\Pi_R(C, G)$. The subproblems are solved by Algorithm RectThreeTypes, which applies $O(|G|)$ flips and transposes by Theorem 4.2. When merging the solutions to the subproblems, we may have to apply Algorithm RectThreeTypes one more time, thus, applying $O(|G|)$ additional flips and transposes. Since each flip or transpose takes $O(1)$ time, the time complexity of the algorithm is $O(|G|)$. □

We now summarize our result in the following theorem.



C
Scanning ($O(mn)$)
$O(mn)$
Partitioning ($O(mn)$)
$O(mn)$
Rotating C' ($O(m'n')$)
Rotating C'' ($O(m''n'')$)
Total $O(m'n') + O(m''n'')$ $= O(mn)$
Algorithm 1 takes $O(m'n')$ time
Algorithm 1 takes $O(m''n'')$ time
Total $O(mn)$
Rotating solution ($O(m'n')$)
Rotating solution ($O(m''n'')$)
Total $O(mn)$
Merging solutions $O(mn)$
$O(mn)$
C_1
Rotating C_1 ($O(mn)$)
$O(mn)$
Algorithm 1 takes $O(mn)$ time
$O(mn)$
Rotating solution ($O(mn)$)
$O(mn)$

Figure 4.13: The steps of Algorithm RectFourTypes, which uses linear $O(|G|)$ time and space.

Theorem 4.5. *Any 1-complex Hamiltonian cycle C of G can be reconfigured to any other 1-complex Hamiltonian cycle in $O(|G|)$ time using $O(|G|)$ flip and transpose operations.*

Proof. In Algorithm RectThreeTypes, the total number of operations required is $O(|G|)$ by Theorem 4.2. Since each flip or transpose operation takes $O(1)$ time, the total time required to scan and perform the operations is $O(|G|)$.

Algorithm RectFourTypes takes $O(|G|)$ time by Theorem 4.4.

In Algorithm CanToCan, we need at most $O(|G|)$ local operations by Theorem 4.1. Since each operation requires $O(1)$ time, the total running time of Algorithm CanToCan is $O(|G|)$. □

4.5 Conclusion

In this chapter, we gave an algorithm to reconfigure any 1-complex Hamiltonian cycle C_1 in a rectangular grid graph to any other cycle C_2 using a linear number of flip and transpose operations. We conclude the chapter with some open problems.

- Is there an algorithm to reconfigure C_1 to C_2 so that the number of flips and transposes used is as small as possible for each input pair?
- Can we achieve similar reconfiguration results for bend complexity $k > 1$?
- What about 1-complex Hamiltonian cycles in d dimension where $d \geq 3$?
- Find other grid graphs that have a splitting track.
- We conjecture that the *splitting* lemma applies to 1-complex cycles in solid grid graphs with an arbitrary number of *reflex corners*.

Chapter 5

1-Complex Cycles in L-Shaped Grid Graphs

In this chapter, we take our research on reconfiguration of 1-complex cycles one step closer to the case of general solid grid graphs, which can have an arbitrary number of *reflex vertices*. Namely, we consider solid grid graphs with a single *reflex vertex*, which we defined as *L-shaped grid graphs* in Chapter 2. Given two 1-complex Hamiltonian cycles C_1 and C_2 of an L-shaped grid graph G, embedded with the top left-corner vertex a at the origin as shown in Figure 5.1, we give an algorithm to reconfigure C_1 to C_2.

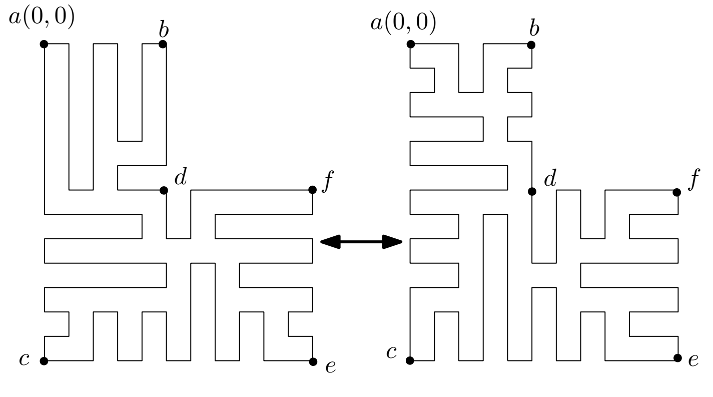


Figure 5.1: A reconfiguration problem on an L-shaped grid graph G: can the first 1-*complex Hamiltonian cycle* be reconfigured to the second by *flips* and *transposes*?

The rest of the chapter is organized as follows.

1. In Section 5.1, we define some terminology that is special to L-shaped grid graphs,

and we adapt some terminology such as the *zip* operation that was defined for cycles in rectangular grid graphs to cycles in L-shaped grid graphs.

2. In Section 5.2, we give algorithms to reconfigure a 1-complex Hamiltonian cycle C_1 to another 1-complex cycle C_2 in an L-shaped grid graph G using $O(|G|)$ flips and transposes, when both C_1 and C_2 have common *forbidden* boundaries, i.e., boundaries that do not contain any cookie *base*.

3. In Section 5.4, we give an algorithm to reconfigure any 1-complex Hamiltonian cycle to any other such cycle of G using $O(|G|)$ flips and transposes, using the algorithms from Section 5.2. In particular, we prove such a reconfiguration *problem* can be split into at most *three subproblems* such that each subproblem, whether rectangular or L-shaped, has a forbidden boundary and hence, can be solved using the algorithms we already know, and that the solutions to the subproblems can be merged into a solution to the original problem in $O(|G|)$ time.

4. In Section 5.5, we discuss some open problems.

5.1 Preliminaries

The terminology for 1-complex Hamiltonian cycles in rectangular grid graphs from Chapter 4 also applies to Hamiltonian cycles in L-shaped grid graphs except for the following. There are *six* types of cookies (W, NE, FE, S, NN, and N) in an L-shaped grid graph as there are six boundaries. The maximum *size* the different types of cookies can have is as follows: $x(d) - 1$ for NE cookies, $x(e) - 1$ for FE cookies, $y(e) - 1$ for NN cookies, $y(e) - y(d) - 1$ for FN cookies, $x(d) - 1$ for W cookies if they are in tracks t_y^h, $1 \leq y < y(d)$, and $x(e) - 1$ when $y(d) < y < y(e) - 2$, and $y(e) - 1$ for S cookies if they are in tracks t_x^v, $1 \leq x < x(d) - 1$, and $y(e) - y(d) - 1$ when $x(d) \leq x < x(e) - 2$.

We say cookies are *from* one of the four axis-aligned *directions*: W cookies from *west*, S from *south*, FN and NN from *north*, and FE and NE from *east*.

In an L-shaped grid graph, we now define canonical Hamiltonian cycles using the term *set of cookies* defined in Chapter 4. See Figure 5.2 for some examples of canonical Hamiltonian cycles. Note that this is not an complete list of canonical cycles. Figures 5.2(a) and (c) show canonical cycles with two sets of cookies from the same direction (west and east, respectively). On the other hand, the canonical cycle in Figure 5.2(b) has two sets of cookies from two different directions: north and east. Figure 5.2(d) shows that a canonical cycle in an L-shaped grid graph can have only one set of cookies.

Definition 5.1 (L-shaped canonical Hamiltonian cycles)**.** *An* L-shaped canonical Hamiltonian cycle *is a* 1*-complex cycle in an L-shaped grid graph that has at most two sets of cookies.*

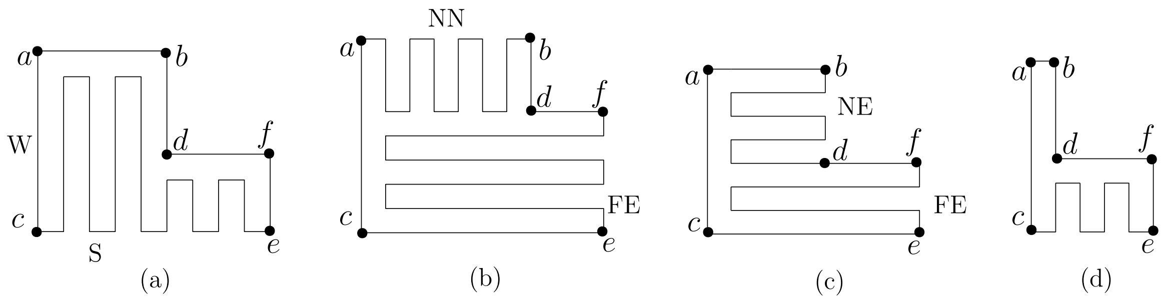


Figure 5.2: Some canonical Hamiltonian cycles in L-shaped grid graphs: (a) two sets of S cookies, which are from the same boundary but have two sizes, (b) a set of NN cookies and a set of FE cookies, (c) a set of NE cookies and a set of FE cookies, and (d) one set of S cookies.

The *zip* operation defined in Chapter 4, along with all related terminology (e.g., *zippability*, *zone of a zip*), for 1-complex cycles in rectangular grid graphs also applies to 1-complex cycles in L-shaped grid graphs.

We now show the use of zippability for 1-complex cycles in L-shaped grid graphs in the following lemma, which is very similar to Lemma 4.1. In fact, the proof of Lemma 4.1 suffices for the following lemma as the proof does not rely on the shape of the outer boundary of the graph as long as the graph is a solid grid graph.

Lemma 5.1. *Let C be a* 1*-complex Hamiltonian cycle in an L-shaped grid graph G and let $Z = zip_{tp}(tr, sz)$ be a zip operation on C, where $tp \in \{W, S, NE, FE, NN, FN\}$. If the zone of Z is zippable, then*

(a) Z reconfigures C to another 1*-complex Hamiltonian cycle C' such that C' has a cookie of type tp and size sz in tr;*

(b) the cycle remains 1*-complex after each transpose and flip operation of Z; and*

(c) in total, at most sz flips and transposes are performed by Z.

Proof. Similar to the proof of Lemma 4.1. □

5.2 Reconfiguring in L-Shaped Grid Graphs: Special Cases

Let C be a 1-complex Hamiltonian cycle of an L-shaped grid graph G. In this section, we consider the following special cases defined by *forbidding* certain cookie types in cycle C:

1. 1-complex cycles without W cookies in Section 5.2.1,

2. 1-complex cycles without any NE or FE cookies in Section 5.2.2, and

3. 1-complex cycles without any NN or FE cookies in Section 5.2.3.

We give algorithms that reconfigure these three special cases of the input cycle to a canonical form. These algorithms are used in Section 5.3 to handle the general case.

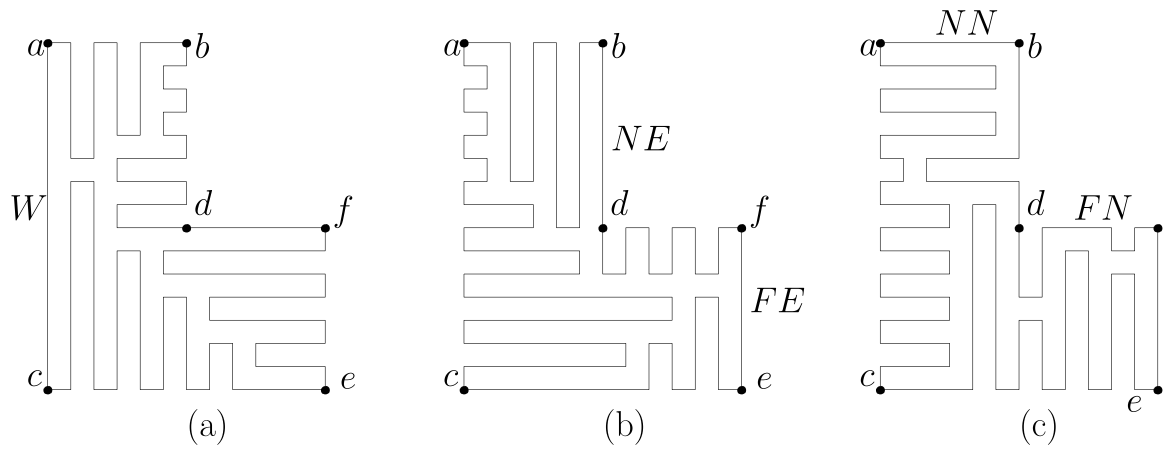


Figure 5.3: Special cases of 1-complex cycles with no cookies of type (a) W, (b) NE or FE, (c) NN or FE.

The algorithms in this section are sweep algorithms whose details depend on parities. The sweeps are designed to ensure that the zippability conditions hold. The algorithms share the same proof of correctness, given in Section 5.2.4.

5.2.1 1-Complex Cycles without W Cookies

Let C be a 1-complex Hamiltonian cycle of G such that W cookies are *forbidden* in C (see Figure 5.3(a)). In this section, we give an algorithm that we call Algorithm XW to reconfigure C to a canonical Hamiltonian cycle $\mathbb{C}$ of G.

Before giving the pseudocode of the algorithm, we give an overview. We consider four cases (see Figure 5.4) based on the parities of $x(d)$ and $x(e)$:

1. *$x(e)$ and $x(d)$ are both odd.* We do one sweep across G from Column 1 to Column $x(e) - 1$ (inclusive), filling alternate vertical tracks with two sets of S cookies of sizes $y(e) - 1$ and $y(e) - y(d) - 1$, respectively.

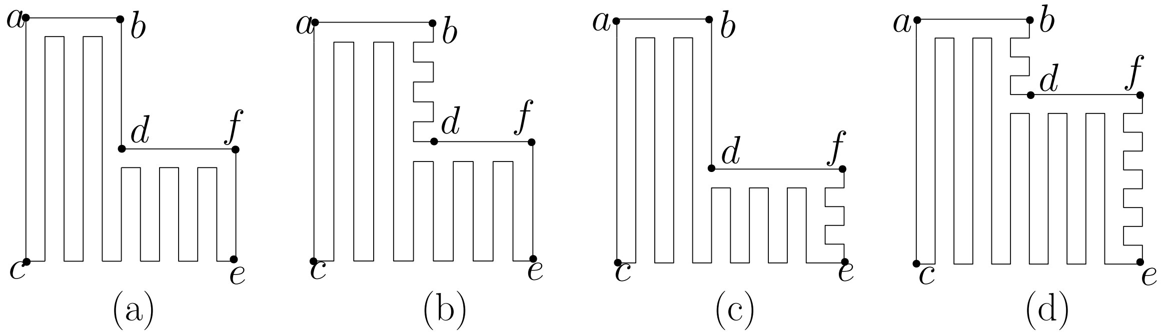


Figure 5.4: The intermediate Hamiltonian cycles after the first sweep in Cases 1–4, respectively. For Case 1, the intermediate cycle is already the final canonical cycle.

2. *$x(e)$ is odd and $x(d)$ is even.* We do two sweeps. The first is from Column 1 to Column $x(e) - 1$ (inclusive), leaving two sets of S cookies, one set with size $y(e) - 1$ and the other with size $y(e) - y(d) - 1$, and also one set of NE cookies of unit size covering the internal vertices of Column $x(d) - 1$ from Row 1 through Row $y(d)$ (inclusive). The second sweep then expands the unit size NE cookies toward the W boundary, namely a downward sweep from Row 1 to Row $y(d)$ (inclusive) fills alternate tracks with NE cookies of size $x(d) - 1$, which shortens the S cookies met to size $y(e) - y(d) - 1$. The final cycle has one set of NE cookies of size $x(d) - 1$ and one set of S cookies of size $y(e) - y(d) - 1$.

3. *$x(e)$ is even and $x(d)$ is odd.* We do two sweeps, similar to Case 2. The first sweep is from Column 1 to Column $x(e) - 1$ (inclusive) and leaves two sets of S cookies, of sizes $y(e) - 1$ and $y(e) - y(d) - 1$, and one set of FE cookies of unit size that covers the internal vertices of Column $x(e) - 1$. The second sweep then expands these unit size FE cookies to Column $x(d)$, namely a downward sweep from Row $y(d) + 1$ to the S boundary partially fills alternate tracks with FE cookies of size $x(e) - x(d)$. The final cycle has one set of FE cookies of size $x(e) - x(d)$ and one set of S cookies of size $y(e) - 1$.

4. *$x(e)$ and $x(d)$ are both even.* We do two sweeps. The first sweep is from Column 1 to Column $x(e) - 1$ (inclusive) and leaves two sets of S cookies, of sizes $y(e) - 1$ and $y(e) - y(d) - 1$, and also one set of NE cookies of unit size that covers the vertices of Column $x(d) - 1$ from Row 1 through Row $y(d)$ (inclusive), and also one set of unit size FE cookies that covers the internal vertices of Column $x(e) - 1$. The second sweep is downward from Row 1 to Row $y(e) - 1$ and expands the unit size NE and FE cookies westward to Column 1.

We now give the pseudocode of the algorithm.

```
**Algorithm** $XW(G, C)$
**Input:** L-shaped grid graph $G$, 1-complex Hamiltonian cycle $C$ of $G$ without
    W cookies
**Output:** A canonical Hamiltonian cycle $\mathbb{C}$ of $G$ without any W cookies.
1.  $i = 1$        //Initialize vertical sweep line
2.  for $i$ from 1 to $x(d) - 1$ ($x(d)$ is odd), or to $x(d) - 2$ ($x(d)$ is even)
3.          $C = zip_S(t_i^v, y(e) - 1)$
4.          $i = i + 2$        //Move sweep line two columns eastward
5.  for $i$ from $x(d)$ ($x(d)$ is odd), or from $x(d) - 1$ ($x(d)$ is even)
6.          to $x(e) - 1$ ($x(e)$ is odd), or to $x(e) - 2$ ($x(e)$ is even)
7.          $C = zip_S(t_i^v, y(e) - y(d) - 1)$
8.          $i = i + 2$        //Move sweep line two columns eastward
9.  if $x(d)$ is even
10.         $j = 1$        //Initialize horizontal sweep line
11.         for $j$ from 1 to $y(d)$
12.                 $C = zip_{NE}(t_j^h, x(d) - 1)$
13.                 $j = j + 2$        //Move sweep line two rows downward
14. if $x(e)$ is even
15.         $j = y(d) + 1$        //Set horizontal sweep line
16.         if $x(d)$ is even
17.                 $FE_h = x(e) - 1$        //Set size of the FE cookies
18.           else  $FE_h = x(e) - x(d)$
19.         for $j$ from $y(d) + 1$ to $y(e) - 1$
20.                 $C = zip_{FE}(t_j^h, FE_h)$
21.                 $j = j + 2$        //Move sweep line two rows downward
22. return $C$
```

5.2.2 1-Complex Cycles without any NE or FE Cookies

Let C be a 1-complex Hamiltonian cycle of G without any NE or FE cookies (see Figure 5.3(b)). We now give an algorithm we call Algorithm XNEFE to reconfigure C to a canonical Hamiltonian cycle.

We first give an overview of the algorithm. We consider four cases based on the parities of $x(d)$ and $x(e)$, as we did in Algorithm XW:

1. *$x(e)$ and $x(d)$ are both odd.* We do one sweep westward from Column $x(e)-1$ to Column 1 (inclusive) that fills alternate vertical tracks with two sets of S cookies of sizes $y(e)-y(d)-1$ and $y(e)-1$, respectively.

2. *$x(e)$ and $x(d)$ are both even.* We do two sweeps. The first sweep is westward from Column $x(e)-1$ to Column 1 (inclusive) and leaves two sets of S cookies, of sizes $y(e)-y(d)-1$ and $y(e)-1$, and also one set of W cookies of unit size that covers the internal vertices of Column 1. The second sweep expands the unit size W cookies eastward to Column $x(d)-1$ (inclusive), namely a downward sweep from Row 1 to Row $y(e)-1$ partially fills alternate horizontal tracks with a set of W cookies of size $x(d)-1$. The final cycle has one set of W cookies of size $x(d)-1$ and one set of S cookies of size $y(e)-y(d)-1$.

3. *$x(e)$ is even and $x(d)$ is odd.* We do two sweeps. The first is westward from Column $x(e)-1$ to Column 1 (inclusive) and leaves one set of S cookies of size $y(e)-y(d)-1$ and one set of NN cookies of size $y(d)$, and also one set of W cookies of unit size that covers vertices of Column 1 from Row $y(d)+1$ to Row $y(e)-1$ (inclusive). The second sweep expands the unit size W cookies eastward to Column $x(e)-1$: a downward sweep from Row $y(d)+1$ to Row $y(e)-1$ partially fills alternate tracks with W cookies of size $x(e)-1$. The final cycle has one set of NN cookies of size $y(d)$ and one set of W cookies of size $x(e)-1$.

4. *$x(e)$ is odd and $x(d)$ is even.* We do two sweeps. The first is westward from Column $x(e)-1$ to Column 1 (inclusive) and leaves one set of S cookies of size $y(e)-y(d)-1$ and one set of NN cookies of size $y(d)$, and also one set of W cookies of unit size. Unlike Case 3, the W cookies cover the vertices of Column 1 from Row 1 to Row $y(d)$. The second sweep expands the unit size W cookies eastward to Column $x(d)-1$: a downward sweep from Row 1 to Row $y(d)$ (inclusive) fills alternate tracks with W cookies of size $x(d)-1$. The final cycle has one set of S cookies of size $y(e)-y(d)-1$ and one set of W cookies of size $x(d)-1$.

We now give the pseudocode of Algorithm XNEFE on the next page.

Algorithm *XNEFE*(G, C)

Input: L-shaped grid graph G, 1-complex Hamiltonian cycle C of G without any NE or FE cookies

Output: A canonical Hamiltonian cycle $\mathbb{C}$ of G without any NE or FE cookies.

```
1.  $i = x(e) - 2$    //Initialize vertical sweep line two columns
2.                    left of the FE boundary
3.  if $x(e) - x(d)$ is even
4.     then for $i$ from $x(e) - 2$ to $x(d)$
5.              $C = zip_S(t_i^v, y(e) - y(d) - 1)$
6.              $i = i - 2$      //Move sweep line two columns westward
7.          for $i$ from $x(d) - 2$ to 2 ($x(d)$ is even) or 1 ($x(d)$ is odd)
8.              $C = zip_S(t_i^v, y(e) - 1)$
9.              $i = i - 2$      //Move sweep line two columns westward
10.         if $x(d)$ is even
11.             for $j$ from 1 to $y(e) - 2$
12.                 $C = zip_W(t_j^h, x(d) - 1)$
13.                 $j = j + 2$  //Move sweep line two rows downward
14.    else
15.         for $i$ from $x(e) - 2$ to 2 ($x(e)$ is even) or 1 ($x(e)$ is odd)
16.             $C = zip_S(t_i^v, y(e) - y(d) - 1)$
17.             $i = i - 2$      //Move sweep line two columns westward
18.         for $i$ from $x(d) - 2$ to 2 ($x(d)$ is even) or 1 ($x(d)$ is odd)
19.             $C = zip_{NN}(t_i^v, y(d))$
20.             $i = i - 2$      //Move sweep line two columns westward
21.         if $x(e)$ is even
22.            then for $j$ from $y(d) + 1$ to $y(e) - 2$
23.                 $C = zip_W(t_j^h, x(e) - 1)$
24.                 $j = j + 2$  //Move sweep line two rows downward
25.            else
26.             for $j$ from 1 to $y(d) - 2$
27.                 $C = zip_W(t_j^h, x(d) - 1)$
28.                 $j = j + 2$  //Move sweep line two rows downward
29. return $C$
```

5.2.3 1-Complex Cycles without any NN or FE Cookies

Let C be a 1-complex Hamiltonian cycle of G without any NN or FE cookies (see Figure 5.3(c)). We give an algorithm called Algorithm XNNFE to reconfigure C to a canonical form.

In this special case, we use the algorithm from the previous section (for cycles without any NE or FE cookies). We sweep downward from Row 1 either to Row $y(d)$ (if $y(d)$ is odd) or to Row $y(d)+1$ (if $y(d)$ is even), and fill alternate horizontal tracks with W cookies of size $x(d)-1$. This removes any initial NE cookies in C. Let C' be the 1-complex Hamiltonian cycle after this sweep. Since C' does not have any NN, NE or FE cookies, we can call Algorithm XNEFE on C' as a procedure.

We now give the pseudocode.

```
Algorithm XNNFE$(G, C)$
Input: L-shaped grid graph $G$, 1-complex Hamiltonian cycle $C$ of $G$ without
    any NN or FE cookies
Output: A canonical Hamiltonian cycle $\mathbb{C}$ of $G$ without any NN or FE
    cookies.
1.  $j = 1$        //Initialize horizontal sweep line
2.  for $j$ from 1 to $y(d)$ ($y(d)$ is odd) or to $y(d) - 1$ ($y(d)$ is even)
3.          $zip_W(t^h_j, x(d) - 1)$
4.          $j = j + 2$        //Move sweep line two rows downward
5.  $C'$ = current 1-complex Hamiltonian cycle.
6.  return Algorithm XNEFE($G$, $C'$)
```

5.2.4 Proof of Correctness

The following theorem establishes the correctness of the Algorithms XW, XNEFE and XNNFE.

Theorem 5.1. *Let C be a 1-complex Hamiltonian cycle in an L-shaped grid graph G. Then Algorithms* XW, XNEFE *and* XNNFE *compute canonical Hamiltonian cycles of G using $O(|G|)$ flips and transposes such that the forbidden cookie types do not appear in the canonical cycles and the cycle remains 1-complex Hamiltonian after each of the operations in the algorithms.*

Proof. Our algorithms consist of a sequence of zips. Note that by Lemma 5.1 we have a 1-complex cycle after each zip operation. Also it is easy to observe that the forbidden cookie types do not appear in any step of the algorithm, and the cycle remains 1-complex Hamiltonian after each operation by Lemma 5.1. We now check zippability of the sweeps.

In the first sweep in all three algorithms, the zone of the first zip in any loop is next to a boundary of G of forbidden cookie type. Thus the line segment halfway between the zone and the boundary can be chosen as the sideline s that does not intersect any cookies. Moreover, any cookies perpendicular to the track cover four internal vertices of G in the zone. Therefore, by Definition 4.5, the zone of the first zip is zippable. The zones of the zip operations that follow in the same loop all have the same size. Thus we can move the sideline by 2 units, and it does not intersect any cookies, and any cookies perpendicular to the zone of the zip would still cover 4 internal vertices of G in the zone. Therefore, the zippability argument applies to all the zips. The end condition of the *for* loops ensures that we never attempt a zip in a track that contains a boundary.

The second sweep (if it is carried out), which is orthogonal to the first sweep, expands cookies of unit size. The zone of the first zip in any loop would have a sideline next to the NN boundary of G or between Rows $y(d)$ and $y(d)+1$, and thus would not intersect any cookies. Furthermore, any cookie perpendicular to the track covers four internal vertices of G in the zone. Therefore, by Definition 4.5, the zone of the first zip is zippable. In a way similar to the first sweep, the zips that follow in the same loop have zones that are zippable.

Since the vertical sweep line sweeps once, performing exactly one zip at each alternate vertical track, at most $O(|G|)$ flips and transposes are performed in the vertical tracks by Lemma 4.1. Similarly, the horizontal sweep line sweeps at most twice, and hence, applies at most two zips in each alternate horizontal track. Then by Lemma 4.1, at most $O(|G|)$ flips and transposes are performed in the horizontal tracks. Therefore, the total number of flips and transposes in each of the three algorithms is $O(|G|)$. $\square$

5.3 Reconfiguring in L-Shaped Grid Graphs: General Case

Let C be a 1-complex Hamiltonian cycle in an L-shaped grid graph G. Now we consider the general case when C has cookies from all four directions (east, west, north, and south); otherwise (possibly after repositioning of G to place vertex e at the origin and vertex a at the bottom right corner) C falls into a special case of Section 5.2.

We first give a formal definition of a "problem" that leads to "subproblems".

Definition 5.2 (Problem $\Pi_L(C, G)$)**.** *Given a* 1*-complex Hamiltonian cycle C in an L-shaped grid graph G, reconfigure C to a canonical Hamiltonian cycle $\mathbb{C}$ of G.*

The L subscript on Π indicates that G is L shaped, just as we used the R subscript in Chapter 4 to indicate that the shape of the grid graph was rectangular there.

The algorithm runs in the three following steps.

1. First it creates *subproblems* of $\Pi_L(C, G)$ in Section 5.3.1.
2. Then it solves the *subproblems* using the algorithms from Section 5.2 in this chapter and also algorithms from Chapter 4; see Section 5.3.2.
3. Finally, it merges the solutions to the *subproblems* to obtain the desired canonical Hamiltonian cycle $\mathbb{C}$ in Section 5.3.3, thus solving $\Pi_L(C, G)$.

5.3.1 Creating Subproblems.

We first recall some terminology from Chapter 4 in order to define *subproblems* of $\Pi_L(C, G)$. We then give an algorithm to create the subproblems.

Definition 5.3 (Splitting Track)**.** *A track t of G is a* splitting track *of C if the interior of t does not intersect any cookies of C and C has cookies on each side of t (i.e., both above and below t, or both on the left and on the right of t).*

We will prove later in Lemma 5.2 that C always has such a splitting track if C has cookies from all four directions (east, west, north and south). Let C have a splitting track t. Without loss of generality assume that t is the vertical track t_i^v. We remove the two edges (u, u') and (v, v') of C at the two ends of t_i^v on boundaries of G, to partition C into two disjoint paths P_{uv} and $P_{u'v'}$. Let the *subgrids* (see Chapter 2) of

G that contain P_{uv} and $P_{u'v'}$ be G' and G'', where G' is to the left of t_i^v and G'' is to the right. Now add a copy of Column i from u to v (inclusive) one unit to the right of G' to create the augmented vertex-induced grid graph $(G')^+$. Note that if $i \geq x(d)$ then $(G')^+$ is L-shaped; otherwise it is rectangular. Without loss of generality assume that $G'^+ = G_L$ is L-shaped. Join u to v by a path through the new vertices to obtain a Hamiltonian cycle C_L of G_L. Similarly obtain a Hamiltonian cycle C_R of $(G'')^+ = G_R$, the vertex-induced grid graph created from G'' by adding a column to its left. The columns added to create $(G')^+$ and $(G'')^+$ are their *augmented boundaries.*

We now formally define the *subproblems* of $\Pi_L(C, G)$. Note that we often say $\Pi_L(C, G)$ *splits into subproblems* although the subproblems are defined on augmented subgrids.

Definition 5.4 (Subproblems)**.** *$\Pi_L(C_L, G_L)$ and $\Pi_R(C_R, G_R)$ are the* subproblems *of $\Pi_L(C, G)$.*

Clearly C_L and C_R have no cookies from the augmented boundaries. Suppose we can solve $\Pi_L(C_L, G_L)$ and $\Pi_R(C_R, G_R)$, obtaining canonical cycles $\mathbb{C}_L$ and $\mathbb{C}_R$, such that $\mathbb{C}_L$ and $\mathbb{C}_R$ have no cookies from the augmented boundaries. Then we can merge $\mathbb{C}_L$ and $\mathbb{C}_R$ by removing the paths along the augmented boundaries and adding back edges (u, u') and (v, v') to obtain a new 1-complex Hamiltonian cycle C_1 of G. We will later show in Section 5.3.3 that C_1 can easily be reconfigured to a canonical form, as C_1 fails to have certain cookie types and falls into one of the special cases handled in Section 5.2.

In addition to the goal of creating subproblems whose solutions are easy to merge, we would also like to achieve a second goal, namely to create subproblems that can be solved with the algorithms of Section 5.2 and of Chapter 4. To achieve these two goals, we choose a splitting track that is either the leftmost vertical splitting track (if it exists) or the southernmost horizontal splitting track. But before finding such a splitting track, we show in the following lemma that a splitting track always exists if C does not fall into the special cases in Section 5.2.

Lemma 5.2. (Splitting track) *Let C be a 1-complex Hamiltonian cycle in an L-shaped grid graph G, where C has cookies from all four directions. Then there is a splitting track tr of G.*

Proof. The proof of this lemma is very similar to the proof of Lemma 4.3. The only difference is that C has cookies from all four *directions* where the precondition in

Lemma 4.3 is that the cycle has cookies from all four *boundaries*. Therefore, we do not repeat the full proof; instead we just prove Event 1 to illustrate the difference.

Event 1. We sweep a vertical line i, column by column, from Column 1 to Column $x(e)-1$ until one of the following two events occurs. We only describe the first event here to show how the proof of Lemma 4.3 can be adjusted to apply here.

1. **Sweep line i intersects cookies from both east and west directions.** Then there must be a pair of cookies from the two directions such that if one of them is in track t^h_j then the other is in t^h_{j+2}, where $1 \leq j < y(e) - 4$. Suppose for a contradiction that the closest pair of cookies c_1 and c_2 from opposite directions are more than two tracks apart. Then some vertices of G on Column i between c_1 and c_2 are not covered by any cookies as no cookies from the north or south directions can cross c_1 or c_2, a contradiction. Therefore, c_1 and c_2 must occupy the tracks t^h_j and t^h_{j+2}, and the track t^h_{j+1} is a splitting track.

2. **Sweep line i does not intersect any W cookies.** We prove this case just as we did in the proof of Lemma 4.3.

□ □

We now give an algorithm we call FindSplitTrack that returns the leftmost vertical splitting track (if it exists) or the southernmost horizontal splitting track of C.

```
Algorithm FindSplitTrack($G, C$)
Input: L-shaped grid graph $G$, 1-complex Hamiltonian cycle $C$ of $G$
Output: A splitting track $t$ of $C$
1.  for every vertical track $t^v_x$ from $x = 2$ up to $x = x(e) - 4$ (inclusive)
2.          if $t^v_x$ has cookies on both sides
3.             then return $t = t^v_x$
4.             else repeat with $x = x + 1$
5.  for every horizontal track $t^h_y$ from $y = y(e) - 4$ down to $y = 2$ (inclusive)
6.          if $t^h_y$ has cookies on both sides
7.             then return $t = t^h_y$
8.             else repeat with $y = y - 1$
```

We prove the correctness of Algorithm FindSplitTrack in the following lemma.

Lemma 5.3. *Let C be a 1-complex Hamiltonian cycle in an L-shaped grid graph G such that C has cookies from all four directions. Then Algorithm* FindSplitTrack *finds in $O(|G|)$ time the leftmost vertical splitting track if it exists, and the southernmost horizontal splitting track otherwise.*

Proof. By Lemma 5.2, a splitting track of C exists. Therefore, if Algorithm FindSplitTrack does not find such a splitting track sweeping eastward, it must find a track when sweeping upward. Let tr be the splitting track returned by the algorithm. It is straightforward to see that tr is either the leftmost vertical splitting track or the southernmost horizontal splitting track. Since there are two sweeps, the time required to find tr is $O(|G|)$. □

Recall that one of the goals of finding a leftmost or a southernmost splitting track (if there is no vertical splitting track) is to make it possible to solve $\Pi_L(C, G)$ by defining subproblems of $\Pi_L(C, G)$ that are solvable by the algorithms we already know from this chapter and the previous chapter. The next algorithm which we call Split generates such subproblems of $\Pi_L(C, G)$.

We now give a description of Algorithm Split and then prove its correctness in the theorem that follows. The algorithm takes a 1-complex Hamiltonian cycle G of an L-shaped grid graph G as input such that C has cookies from all four directions, and creates at most three subproblems of $\Pi_L(C, G)$. We first run the algorithm FindSplitTrack(G, C) on C. Let tr be the splitting track of C returned by Algorithm FindSplitTrack. We remove edges (u, u') and (v, v') of C at the two ends of tr to partition C into disjoint paths P_{uv} and $P_{u'v'}$. Let the subgrids of G containing P_{uv} and $P_{u'v'}$ be G' and G''. We augment vertices to G' and G'' to get $(G')^+$ and $(G'')^+$, respectively. Let $(G')^+$ be L-shaped and $(G'')^+$ be rectangular, without loss of generality. We then connect u to v and u' to v' through the new vertices to get 1-complex Hamiltonian cycles C_L and C_R of $(G')^+$ and $(G'')^+$, respectively. If C_L does not have cookies from all four directions then $\Pi_R(C_R, (G'')^+)$ and $\Pi_L(C_L, (G')^+)$ are the two desired subproblems of $\Pi_L(C, G)$, and the algorithm ends.

If C_L has cookies from all four directions, we repeat the above process again to obtain subproblems $\Pi_R(C'_R, (G^i)^+)$ and $\Pi_L(C'_L, (G^{ii})^+)$ of $\Pi_L(C_l, (G')^+)$. Then the three subproblems of $\Pi_L(C, G)$ are $\Pi_R(C_R, (G')^+)$, $\Pi_R(C'_R, (G^i)^+)$, and $\Pi_L(C'_L, (G^{ii})^+)$.

We now prove the correctness of the above algorithm.

Theorem 5.2. *Let C be a 1-complex cycle of an L-shaped grid graph G such that C has cookies from all four directions. Then Algorithm* Split *creates from $\Pi_L(C, G)$*

in $O(|G|)$ time either the subproblems $\Pi_R(C_R, (G')^+)$ and $\Pi_L(C_L, (G'')^+)$, or the subproblems $\Pi_R(C_R, (G')^+)$, $\Pi_L(C'_L, (G^{ii})^+)$ and $\Pi_R(C'_R, (G^i)^+)$; here each of C_R and C'_R has at most three types of cookies, and each of C_L and C'_L falls into one of the cases of Section 5.2.

Proof. We first assume the algorithm returns two subproblems $\Pi_R(C_R, (G')^+)$ and $\Pi_L(C_L, (G'')^+)$ as shown in Figures 5.5(a)–(b). It is easy to observe that C_R has at most three types of cookies since the augmented boundary of $(G'')^+$ does not have cookies. Since C_L does not have cookies from all four directions it must fall into one of these following special cases.

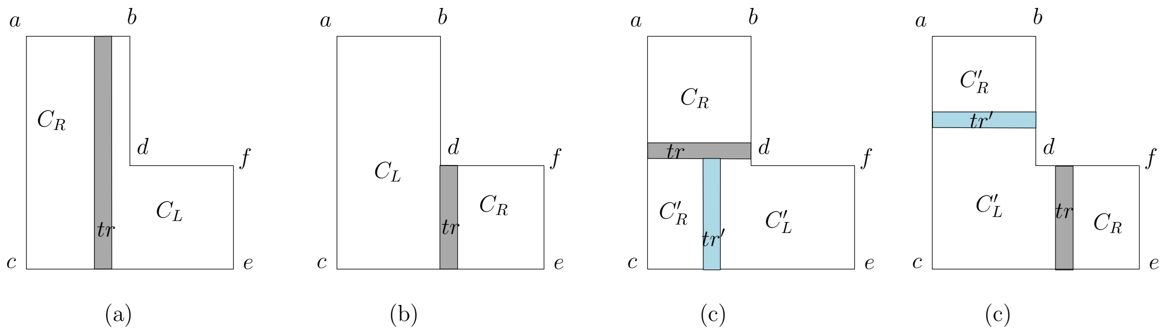


Figure 5.5: (a) and (b) $\Pi_L(C, G)$ is split into two subproblems (the splitting track tr is shown in gray). (c) $\Pi_L(C, G)$ is split into three subproblems. The two splitting tracks tr (gray) and tr' (blue) are perpendicular and intersect each other, and (d) the splitting tracks tr and tr' are perpendicular to each other but do not intersect.

1. No W cookies. See Section 5.2.1.
2. No S cookies. It is the same case as having W cookies forbidden subject to repositioning of G to place vertex e at the origin and vertex a at the bottom right corner.
3. No NE and FE cookies. See Section 5.2.2.
4. No NN and FN cookies. Same as Case 3 subject to repositioning of G.

We now assume that three subproblems are returned: $\Pi_R(C_R, (G')^+)$, $\Pi_L(C'_L, (G^{ii})^+)$ and $\Pi_R(C'_R, (G^i)^+)$. If tr is horizontal then tr' must be vertical and vice versa. Assume for a contradiction that both tr and tr' are horizontal. Since tr' is closer to the south boundary than tr, FindSplitTrack (G, C) cannot return tr instead of tr', a contradiction. Similarly we can show that both the splitting tracks cannot be vertical either.

We have two cases to consider based on the orientation of tr.

1. tr is horizontal. Then it must be above Row $y(d)$ and vertical track tr' must intersect tr as shown in Figure 5.5(c). Then C_L does not have any W or NN cookies and hence is the special case of Section 5.2.1, and C'_R has at most two types of cookies.

2. tr is vertical. Then it must be to the right of Column $x(d)$. If the horizontal splitting track tr' intersects tr below Row $y(d)$ then the case is similar to the previous case. Otherwise tr' is above Row $y(d)$ as shown in Figure 5.5(d). Then C_L does not have any NN or FE cookies and hence is the special case of Section 5.2.3, and C'_R has at most three types of cookies.

□

5.3.2 Solving the Subproblems.

We first show how to solve the rectangular subproblems, and then show how to solve the L-shaped subproblem.

Rectangular Subproblem(s)

To solve $\Pi_R(C_R, (G')^+)$ (and $\Pi_R(C'_R, (G^i)^+)$ if $\Pi_L(C, G)$ is split into three subproblems), we apply the algorithm for rectangular 1-complex cycles with no E cookies from Chapter 4 to get canonical forms $\mathbb{C}_R$ (and $\mathbb{C}'_R$), where the canonical cycle has either a set of W cookies or a set of N cookies.

Since C_R has exactly one augmented boundary, we position C_R such that the augmented boundary is the E boundary, as there are no cookies from that boundary. However, in case $\Pi_L(C, G)$ is split into three subproblems and $\Pi_R(C'_R, (G^i)^+)$ exists, C'_R can have two perpendicular augmented boundaries (see Figure 5.5(c)). In that case, we position C'_R such that the augmented boundaries are its E and S boundaries, so that the $\mathbb{C}'_R$ produced by the algorithm does not have cookies from the augmented boundaries.

L-Shaped Subproblem

Since C_L falls into one of the special cases of Section 5.2 (possibly after repositioning G) by Theorem 5.2, we apply the appropriate algorithm to get a canonical form $\mathbb{C}_L$ from C_L that does not have cookies from the augmented boundaries.

5.3.3 Merging the Solutions to the Subproblems.

We first assume that $\Pi_L(C, G)$ is split into two subproblems by splitting track tr and show how to merge the solutions to $\Pi_R(C_R, (G')^+)$ and $\Pi_L(C_L, (G'')^+)$. We then show how to merge the solutions to the subproblems when $\Pi_L(C, G)$ is split into three subproblems.

The case of two subproblems

We merge the solutions $\mathbb{C}_L$ of $\Pi_L(C_L, (G'')^+)$ and $\mathbb{C}_R$ of $\Pi_R(C_R, (G')^+)$ output by the algorithms as follows. We remove the augmented boundaries from $\mathbb{C}_L$ and $\mathbb{C}_R$ to get two Hamiltonian paths in the original subgrids G' and G'' of G on both sides of the splitting track tr. We join the paths using the boundary edges at the ends of track tr to obtain a 1-complex Hamiltonian cycle C_1 of G. Cycle C_1 has at most three sets of cookies as $\mathbb{C}_L$ and $\mathbb{C}_R$ have at most two sets and one set of cookies, respectively. Thus C_1 cannot have cookies from all four directions and must fall into a special case of Section 5.2. We then apply the appropriate algorithm on C_1 to obtain a canonical Hamiltonian cycle $\mathbb{C}$ of G. See Figure 5.6.

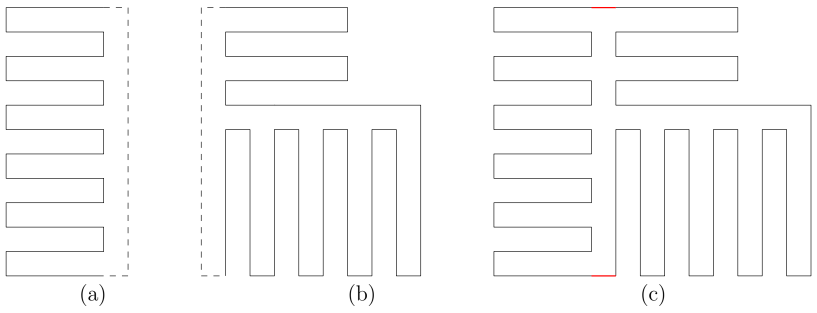


Figure 5.6: (a) Solution $\mathbb{C}_R$ of subproblem $\Pi_R(C_R, (G')^+)$, and (b) solution $\mathbb{C}_L$ of subproblem $\Pi_L(C_L, (G'')^+)$ of $\Pi_L(C, G)$. The augmented boundaries are shown in dashed lines. (c) Hamiltonian cycle of G after removing the augmented boundaries from $\mathbb{C}_R$ and $\mathbb{C}_L$ and joining the paths using the edges on the boundary of G that were removed to create $\Pi_R(C_R, (G')^+)$ and $\Pi_L(C_L, (G'')^+)$.

The case of three subproblems

We now assume that $\Pi_L(C, G)$ is split into three subproblems $\Pi_R(C_R, (G')^+)$, $\Pi_L(C'_L, (G^{ii})^+)$ and $\Pi_R(C'_R, (G^i)^+)$, first by the splitting track tr and then by the splitting track tr'. The idea is to merge the subproblems created by the second splitting track (tr') first. Therefore, we first merge the canonical forms $\mathbb{C}'_L$ and $\mathbb{C}'_R$, which are solutions to $\Pi_L(C'_L, (G^{ii})^+)$ and $\Pi_R(C'_R, (G^i)^+)$, respectively, in the same way shown above to obtain $\mathbb{C}_L$. Then we merge $\mathbb{C}_L$ with $\mathbb{C}_R$, which is the solution to the subproblem $\Pi_R(C_R, (G')^+)$ created by the first splitting track tr, to obtain $\mathbb{C}$ of G.

We now summarize our result from this section in the following theorem.

Theorem 5.3. *Let C be a 1-complex Hamiltonian cycle of an L-shaped grid graph G. Then $\Pi_L(C, G)$ can be solved using $O(|G|)$ flips and transposes.*

Proof. By Theorem 5.2, Split creates at most three subproblems, with $O(|G|)$ flips and transposes. Each of the subproblems is solved by an algorithm that applies $O(|G|)$ flips and transposes as seen above. It is easy to see that the merging of the solutions to the subproblems makes $O(|G|)$ flips and transposes. □

5.4 Reconfiguration between any Pair of 1-Complex Cycles

The previous section discussed the problem of reconfiguring any 1-complex Hamiltonian cycle C of an L-shaped grid graph G to a canonical cycle. Here we generalize the problem by removing the constraints on the goal, i.e., the problem is now reconfiguring C to any other 1-complex Hamiltonian cycle of G that is not necessarily canonical.

We first give a formal definition of the problem.

Definition 5.5 (Problem $\Pi_L(C_1, C_2, G)$)**.** *Given any pair of 1-complex Hamiltonian cycles C_1 and C_2 of an L-shaped grid graph G, reconfigure C_1 to C_2 using flips and transposes.*

To solve $\Pi_L(C_1, C_2, G)$, we first solve $\Pi_L(C_1, G)$ and $\Pi_L(C_2, G)$ yielding $\mathbb{C}_1$ and $\mathbb{C}_2$. Now we give an algorithm for $\Pi_L(\mathbb{C}_1, \mathbb{C}_2, G)$, a special case of $\Pi_L(C_1, C_2, G)$. In order to do that we observe some properties special to canonical Hamiltonian cycles of L-shaped grid graphs. We then use those properties to split $\Pi_L(\mathbb{C}_1, \mathbb{C}_2, G)$ into two rectangular subproblems.

Let G be an L-shaped grid graph. By definition, any canonical cycle $\mathbb{C}$ of G has at most two sets S_1 and S_2 of cookies, where the cookies of each set are of the same type and size. If there are exactly two sets of cookies then there is a unique track tr between the rectangular regions covered by S_1 and S_2. Note that tr must be a splitting track of $\mathbb{C}$. We call tr the *canonical splitting track* of $\mathbb{C}$. It is easy to observe that tr is either $t^v_{x(d)-1}$ (Figure 5.7(a)) or $t^h_{y(d)}$ (Figures 5.7(b)–(c)).

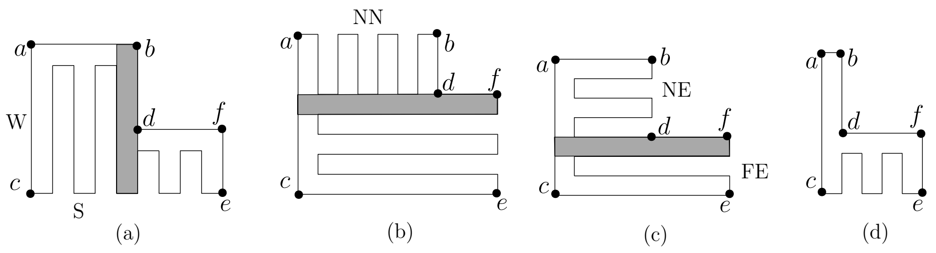


Figure 5.7: Some canonical Hamiltonian cycles (canonical splitting tracks shown grey).

Let $\mathbb{C}_1$ and $\mathbb{C}_2$ have the same canonical splitting track $tr = t^v_{x(d)-1}$. We remove the edges of $\mathbb{C}_1$ in tr that are on boundaries of G (i.e., NN, NE and S boundaries) to partition $\mathbb{C}_1$ into two disjoint paths, P' from $v_{x(d)-1,0}$ to $v_{x(d)-1,y(e)}$ to the left of tr and P'' from d to $v_{x(d),y(e)}$ to the right of tr. Let G' and G'' be the rectangular subgrids of G that contain P' and P''. We generate the augmented vertex-induced grid graphs $(G')^+$ and $(G'')^+$ by adding a copy of Column $x(d) - 1$ one unit to the right of G' and a copy of Column $x(d)$ from d to $v_{x(d),y(e)}$ (inclusive) to the left of G'', respectively. Note that both $(G')^+$ and $(G'')^+$ are rectangular. We then join the two endpoints of P' through the new vertices of $(G')^+$ and the endpoints of P'' through the new vertices of $(G'')^+$ to obtain canonical Hamiltonian cycles $\mathbb{C}'_1$ of $(G')^+$ and $\mathbb{C}''_1$ of $(G'')^+$. In a similar way we obtain canonical Hamiltonian cycles $\mathbb{C}'_2$ and $\mathbb{C}''_2$ from $\mathbb{C}_2$ such that $\mathbb{C}'_1$ covers the same rectangular subgrid of G that $\mathbb{C}'_2$ covers, and such that $\mathbb{C}''_1$ covers the same rectangular subgrid of G that $\mathbb{C}''_2$ covers. We say that $\Pi_R(\mathbb{C}'_1, \mathbb{C}'_2, (G')^+)$ and $\Pi_R(\mathbb{C}''_1, \mathbb{C}''_2, (G'')^+)$ are the two *subproblems* of $\Pi_L(\mathbb{C}_1, \mathbb{C}_2, G)$.

We now show that the problem $\Pi_L(\mathbb{C}_1, \mathbb{C}_2, G)$ can be solved by solving its two subproblems.

Theorem 5.4. *Let $\mathbb{C}_1$ and $\mathbb{C}_2$ be any canonical Hamiltonian cycles of an L-shaped grid graph G. Then $\Pi_L(\mathbb{C}_1, \mathbb{C}_2, G)$ can be solved with $O(|G|)$ flips and transposes.*

Proof. We consider two cases.

1. Both $\mathbb{C}_1$ and $\mathbb{C}_2$ have the same canonical splitting track tr. We partition $\Pi_L(\mathbb{C}_1, \mathbb{C}_2, G)$ into two subproblems $\Pi_R(\mathbb{C}_1', \mathbb{C}_2', (G')^+)$ and $\Pi_R(\mathbb{C}_1'', \mathbb{C}_2'', (G'')^+)$, solve the subproblems using Algorithm CanToCan from Chapter 4, and merge the solutions by removing the augmented edges from $\mathbb{C}_2'$ and $\mathbb{C}_2''$ and rejoining the paths with all the edges in track tr on the boundary of G previously removed to create the subproblems.

2. Assume $\mathbb{C}_1$ and $\mathbb{C}_2$ have $t^v_{x(d)-1}$ and $t^h_{y(d)}$ as canonical splitting tracks, respectively. As a subgoal, we reconfigure $\mathbb{C}_1$ to another canonical Hamiltonian cycle $\mathbb{C}_3$ of G such that $\mathbb{C}_3$ and $\mathbb{C}_2$ have the same canonical splitting track $t^h_{y(d)}$.

   If $x(d)$ is odd, we apply transposes and/or flips on $\mathbb{C}_1$ to cover the columns to the left of $t^v_{x(d)-1}$ by a set of NN cookies of size $y(d)$ and a set of S cookies of size $y(e) - y(d) - 1$. If there is a set of S cookies to the right of track $t^v_{x(d)-1}$ for $\mathbb{C}_1$ then we have a canonical cycle with splitting track $t^h_{y(d)}$ as desired. If $\mathbb{C}_1$ had FN cookies to the right of $t^v_{x(d)-1}$, then we apply flips to reconfigure them to a set of S cookies of height $y(e) - y(d) - 1$, and thus we get the canonical cycle with splitting track $t^h_{y(d)}$. Otherwise, in $\mathbb{C}_1$, there must be a set of FE cookies right of $t^v_{x(d)-1}$, which we extend to Column 1 to get the canonical cycle with splitting track $t^h_{y(d)}$.

   If $x(d)$ is even, then $y(d)$ must be even, and the columns to the left of $t^v_{x(d)-1}$ in $\mathbb{C}_1$ must be covered by W cookies. If we have S or FN cookies to the right of $t^v_{x(d)-1}$, we extend the W cookies below track $t^h_{y(d)}$ to Column $x(e)-1$ by applying transposes. Otherwise the cookies to the right of $t^v_{x(d)-1}$ must be FE, and we extend the W cookies below track $t^h_{y(d)}$ to Column $x(e) - 1$ by applying flips. In this way we obtain the desired canonical Hamiltonian cycle $\mathbb{C}_3$ with splitting track $t^h_{y(d)}$. We then solve $\Pi_L(\mathbb{C}_3, \mathbb{C}_2, G)$ as done previously, which gives a solution to $\Pi_L(\mathbb{C}_1, \mathbb{C}_2, G)$.

□

We now summarize our main reconfiguration result from this section.

Theorem 5.5. *Let C_1 and C_2 be any two 1-complex Hamiltonian cycles of an L-shaped grid graph G. Then $\Pi_L(C_1, C_2, C)$ can be solved with $O(|G|)$ flips and transposes such that the cycle remains 1-complex Hamiltonian after each operation.*

Proof. By Theorem 5.3, C_1 and C_2 can be reconfigured to canonical cycles $\mathbb{C}_1$ and $\mathbb{C}_2$ of G using $O(|G|)$ flips and transposes in total. By Theorem 5.4, $\Pi_L(\mathbb{C}_1, \mathbb{C}_2)$ can be solved also using $O(|G|)$ flips and transposes. By Lemma 5.1, the cycle remains 1-complex Hamiltonian after each operation. □

5.5 Conclusion

In this chapter we showed that any 1-complex Hamiltonian cycle C_1 of an L-shaped grid graph G can be reconfigured to any other 1-complex Hamiltonian cycle C_2 of G. We conclude with the following open problems.

1. We adapted the *zip* operation defined in Chapter 4 for 1-complex cycles in rectangular grid graphs for L-shaped grid graphs. Is it possible to generalize zip to solid grid graphs when the cycle is 1-complex?

2. We conjecture that the Splitting Lemma (Lemma 5.2) applies to any solid grid graph with an arbitrary orthogonal polygonal boundary. The reason behind that is that we proved that a splitting track exists when the cycle has cookies from all four *directions*, which does not rely on the shape of the boundary of the graph. But it would be nice to actually prove it.

3. For 1 reflex corner (the L-shaped case), we had to split a Hamiltonian cycle at most twice. Is it possible to split 1-complex cycles in any solid grid graph into a number of subproblems linear in the number of reflex vertices, such that each of the subproblems is *solvable*?

4. More generally, how does the number of subproblems created by a splitting approach (when possible) depend on the number of reflex corners?

Chapter 6

Hamiltonian Paths in Rectangular Grid Graphs

In this chapter, we discuss 1-complex s, t Hamiltonian paths between the top-left and bottom-right corners of a rectangular grid graph, and reconfiguration of such paths.

In a Hamiltonian cycle in a grid graph, each internal vertex is covered by some *cookie*, i.e., a subpath that starts and ends on adjacent vertices on the same boundary and contains no vertices on the boundary other than the endpoints. In contrast to that, Hamiltonian paths cover internal vertices of the grid with cookies as well as with subpaths that we call "separators" (defined in Section 6.1). Unlike cookies, separators have endpoints on different boundaries and can contain either one, or two, or no internal bends. Because of this additional feature in the structure of s, t Hamiltonian paths, we need some new strategies in addition to the strategies we used for cycles in Chapters 4 and 5.

The chapter outline is as follows.

1. In Section 6.1, we define some terminology that is special to Hamiltonian paths.

2. In Section 6.2, we establish the structure of 1-complex s, t Hamiltonian paths in rectangular grid graphs by dividing such paths into subpaths and establishing the structure of each subpath separately.

3. In Section 6.3, we give a research plan to obtain algorithms to reconfigure any 1-complex s, t Hamiltonian path P_1 in a rectangular grid graph to any other such path P_2 using flips, transposes, and switches, based on the structure of Hamiltonian paths that we establish in Section 6.2. We also make use of results for

reconfiguration of cycles from Chapters 4 and 5 in designing algorithms for paths.

4. Section 6.4 collects together some open problems arising from the chapter.

6.1 Separators and Their Properties

Let P be a directed 1-complex Hamiltonian path in an $m \times n$ grid graph G from the upper left corner s to the bottom right corner t. We now give some terminology and observe some properties of P. In particular we define and examine the properties of "separators", which are subpaths of P whose removal from G leaves s and t in separate components of G.

The internal (non-boundary) vertices of G are each covered by some subpath P' of P such that P' contains at least one internal vertex of G and intersects the boundary of G precisely at the two endpoints of P'. If the endpoints lie on the same boundary (N, S, E, or W) we call the subpath P' a *cookie* of P. Note that in a cookie, P' must contain at least two internal vertices of G as otherwise P' cannot have both endpoints on the same boundary. If P' has its two endpoints on two different boundaries then we call P' a *separator* of P.

Definition 6.1 (Separator). *A* separator *of P in G is a subpath P' of P that starts and ends on different boundaries of G, and that covers at least one internal vertex of G, and that contains no boundary vertices other than the endpoints of P'.*

We denote by $P_{u,v}$ a directed subpath of P that starts at u and ends at v, where u is closer to s than v when following P from s to t. Hence $P_{s,t}$ denotes the directed path P from s to t.

We have the following observation.

Observation 6.1. *Any separator of P in G has* 0, 1, *or* 2 *internal bends, and if there are two internal bends, they must be adjacent.*

Proof. (See Figure 6.1.) Let $P_{u,v}$ be a separator of P where u and v are vertices on two different boundaries of G and suppose $P_{u,v}$ contains at least one bend. Tracing $P_{u,v}$ from u to the first internal bend a, we consider the vertex b we encounter next after a. Since $P_{u,v}$ is 1-complex, b must be connected to v by a straight line segment on P. Hence if $P_{u,v}$ does not bend at b, then $P_{u,v}$ has no further bends, and if $P_{u,v}$ does bend at b then $P_{u,v}$ has one of the forms shown in Figure 6.1(c). □

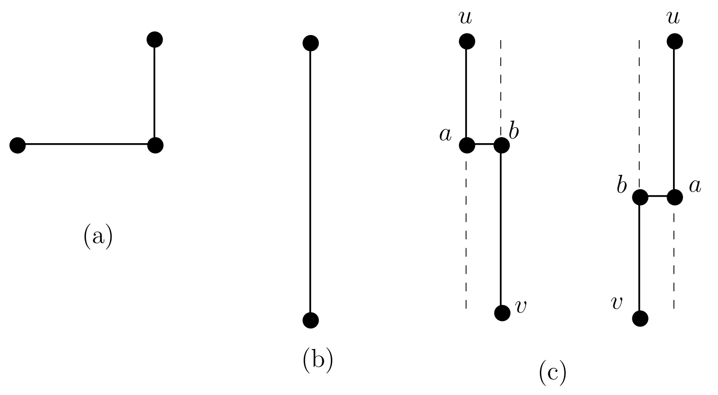


Figure 6.1: Separators with (a) one bend, (b) no bends, (c) two bends.

Based on the number of bends, we define three kinds of separators as follows.

1. A *corner separator* contains exactly one bend (see Figure 6.1(a)).
2. A *straight separator* contains no bends (see Figure 6.1(b)).
3. A *bent separator* contains exactly two bends as shown in Figure 6.1(c).

We call a corner separator a *corner cookie* if P connects one of the endpoints of the separator to s or t by a straight subpath lying on a boundary of G. Figure 6.2 shows examples of corner cookies. From now on, we consider a corner cookie to be a cookie, not a separator.

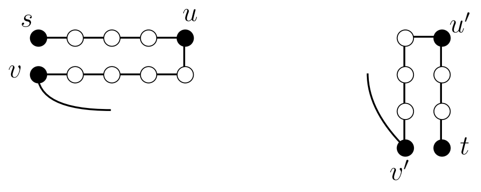


Figure 6.2: The corner separators $P_{u,v}$ and $P_{v',u'}$ are regarded as corner cookies.

We now establish properties of different kinds of separators that we use in the next section to establish the structure of P.

Let $P_{s,t}$ be the 1-complex s, t Hamiltonian path shown in Figure 6.3. Let α and β denote the bottom left and top right corner vertices of G. Since we can fix s and reflect G about the line $x = y$, without loss of generality, from now on we make the following assumption.

Assumption: We assume that $P_{s,t}$ visits α before visiting β.

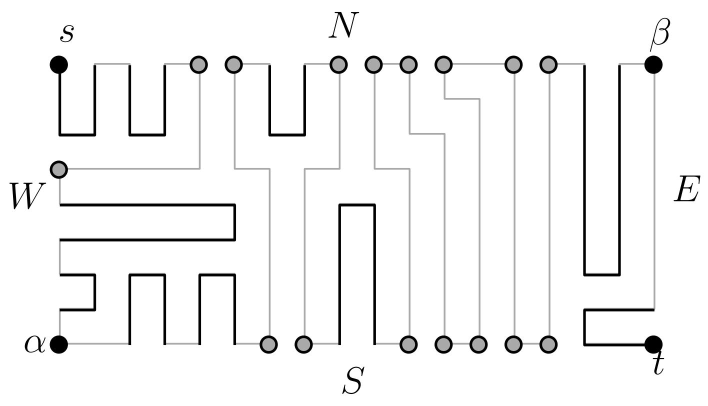


Figure 6.3: A 1-complex s, t Hamiltonian path $P_{s,t}$. Cookies of $P_{s,t}$ are shown in black and the endpoints of the separators of $P_{s,t}$ are shown in gray circles.

We now have the following observations.

Observation 6.2. *The subpath $P_{s,\alpha}$ covers all the W vertices and the subpath $P_{\beta,t}$ covers all the E vertices.*

Proof. If the subpath $P_{s,\alpha}$ does not cover all the W vertices, $P_{s,t}$ fails either to be Hamiltonian or to be non-crossing. Therefore, $P_{s,t}$ must visit all the W vertices before reaching α. By similar logic, $P_{\beta,t}$ must cover all the E vertices. □

Corner separators have endpoints on boundaries sharing a common corner. We say that a corner separator *cuts off* this common corner.

Observation 6.3. *All corner separators cutting off s occur before $P_{s,t}$ reaches α, and no corner separators cut off α or β; all corner separators cutting off t occur after $P_{s,t}$ reaches β.*

Proof. All the corner separators cutting off s have one endpoint on the W boundary. By Observation 6.2, all the W vertices must be visited by $P_{s,t}$ before $P_{s,t}$ reaches α. Therefore, all the corner separators cutting off s must occur before $P_{s,t}$ reaches α. Similarly, we can prove that all the corner separators cutting off t must occur after $P_{s,t}$ visits β.

Suppose there is a corner separator cutting off α. Then $P_{s,t}$ is either not Hamiltonian or crosses itself, which is a contradiction to $P_{s,t}$ being an s, t Hamiltonian path of G. So no corner separator that cuts off α exists. Similarly, there cannot be any corner separator cutting off β. □

Traveling along $P_{s,t}$, we denote the i-th corner separator that cuts off s by μ_i, and the i-th corner separator that cuts off t by ν_i. The start and end points of μ_i and ν_i are denoted by $s(\mu_i), t(\mu_i)$ and $s(\nu_i), t(\nu_i)$, respectively, and the internal bends are denoted by $b(\mu_i)$ and $b(\nu_i)$, respectively.

Since the endpoints of a straight or a bent separator lie on opposite boundaries (N and S, or E and W), we make the following definition.

Definition 6.2 (Cross Separator). *We call a straight or bent separator a* cross separator.

Our notation for cross separators is similar to that for corner separators. Traveling along $P_{s,t}$, we denote the i-th cross separator we encounter by η_i; we denote the endpoints of η_i by $s(\eta_i)$ and $t(\eta_i)$, where $s(\eta_i)$ is the endpoint closer to s along $P_{s,t}$. If η_i is a bent separator, then we let $b(\eta_i)$ denote the internal bend of η_i that has the same x-coordinate as $s(\eta_i)$. Figure 6.4 shows the endpoints and internal bends of all the separators of the Hamiltonian path in Figure 6.3, and Figure 6.5 gives a combinatorial layout of the same Hamiltonian path.

We now observe properties of *cross separators.*

Lemma 6.1. *Let $P_{s,t}$ visit the lower left corner α of G before the upper right corner β of G. Then the following statements hold.*

(a) All the cross separators must have endpoints on the N and S boundaries of G.

(b) All the cross separators of $P_{s,t}$ must occur between α and β.

(c) There is at least one cross separator, and the first and last cross separators, which may be the same, start on the S boundary and end on the N boundary of G.

(d) There must be an odd number $k \geq 1$ of cross separators in $P_{s,t}$.

Proof. (a) Since $P_{s,t}$ is non-crossing, all the cross separators have endpoints on the same pair of opposite boundaries (either N and S, or E and W). If the cross separators have endpoints on the E and W boundaries then $P_{s,t}$ cannot visit α before β as we assumed, since otherwise path $P_{s,t}$ fails to be Hamiltonian or non-crossing. Therefore, all the cross separators must have endpoints on the N and S boundaries.

By Observation 6.3, $P_{\alpha,\beta}$ must leave α along the S boundary and reach β along the N boundary. Therefore, $P_{\alpha,\beta}$ contains a cross separator that travels from the S to the N boundary.

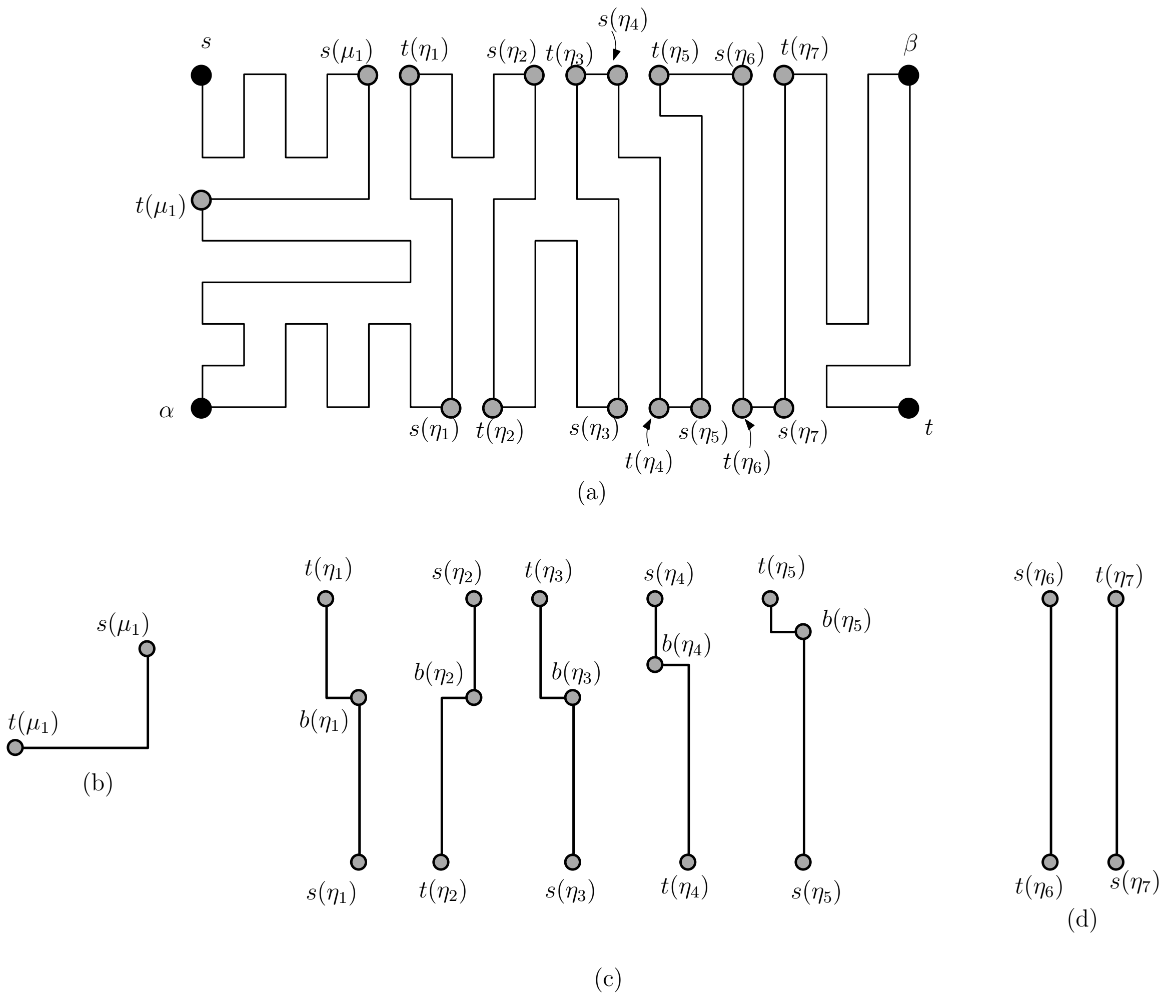


Figure 6.4: (a) The separators of the path $P_{s,t}$ of Figure 6.3. (b) The corner separator, (c) the bent separators, and (d) the straight separators, shown separately with their endpoints and bends marked.

(b) Since by Observation 6.2, all the W vertices must occur in $P_{s,\alpha}$ and all the E vertices must be covered by subpath $P_{\beta,t}$, the starting point $s(\eta_1)$ of the first cross separator η_1 must occur after α but before β on $P_{s,t}$. Similarly, the ending point $t(\eta_k)$ of η_k must occur before β. Therefore, all the cross separators must occur between α and β on $P_{s,t}$. See Figure 6.5.

(c) If $s(\eta_1)$ is on the N boundary then $P_{s,t}$ either is not Hamiltonian or crosses itself, a contradiction. Therefore $s(\eta_1)$ must be on the S boundary and $t(\eta_1)$ must be on the N boundary. Similarly, letting η_k, $k \geq 1$, denote the last cross separator of $P_{\alpha,\beta}$, if $t(\eta_k)$ is on the S boundary then $P_{s,t}$ cannot visit β and hence cannot

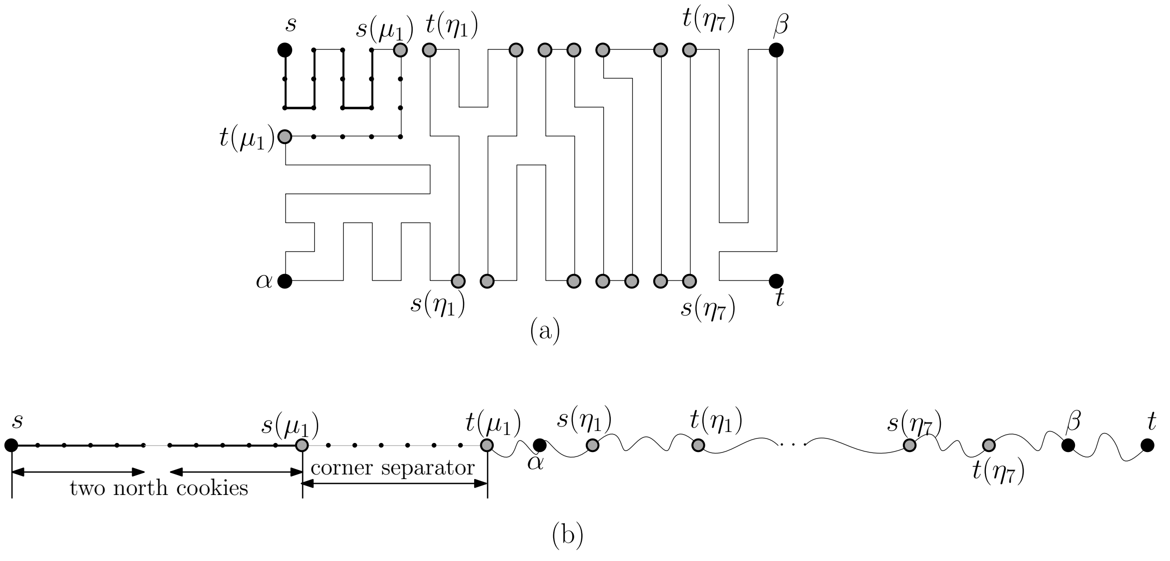


Figure 6.5: (a) The path $P_{s,t}$ from Figure 6.3. The vertices on the two N cookies and the first corner separator μ_1 along $P_{s,s(\mu_1)}$ are shown in small black circles. (b) A combinatorial layout of the path in (a).

be Hamiltonian. Therefore, t_{η_k} must be on the N boundary and $s(\eta_k)$ must be on the S boundary.

(d) The first cross separator η_1 starts on the S boundary and ends on the N boundary. Since $P_{s,t}$ cannot cross itself, $s(\eta_2)$ must be on the N boundary and $t(\eta_2)$ must be on the S boundary. In this way $s(\eta_i)$ must be on the S boundary when i is odd and $s(\eta_i)$ must be on the N boundary when i is even. Since $s(\eta_k)$ must be on the S boundary by Claim (c), k must be odd.

□

6.2 Structure of 1-Complex Hamiltonian Paths

In this section, we establish the structure of $P_{s,t}$. In order to do that, we break $P_{s,t}$ into *three subpaths* such that two consecutive subpaths overlap at their common endpoint. We then establish the structures of the three subpaths separately.

We first define the *three subpaths* of $P_{s,t}$.

Definition 6.3 (Initial Subpath)**.** *The* initial subpath *of $P_{s,t}$ is the subpath $P_{s,s(\eta_1)}$.*

Definition 6.4 (Middle Subpath)**.** *The* middle subpath *of $P_{s,t}$ is the subpath $P_{s(\eta_1),t(\eta_k)}$.*

Definition 6.5 (Final Subpath)**.** *The* final subpath *of $P_{s,t}$ is the subpath $P_{t(\eta_k),t}$.*

We establish the structures of the three subpaths of $P_{s,t}$ in the following order. We first discuss the initial and final subpaths of $P_{s,t}$ since they have similar structures. We then discuss the middle subpath of $P_{s,t}$.

To establish the structure of the initial subpath of $P_{s,t}$, we define some terminology. A *sequence q of E (or W) cookies* is a group of E (or W) cookies that are connected consecutively on $P_{s,t}$ by single edges on the E (or W) boundary; thus, as a group the cookies cover vertices in adjacent rows in Column $n-2$ (Column 1) of G. See Figure 6.6(a) and (b). A *tabletop* of q is a maximal consecutive subsequence of E (or W) cookies of maximum cookie size in q. The sequence q is *unimodal* if it has just one tabletop and the sizes of the cookies are non-increasing as we move further along P from the tabletop. Figures 6.6(a)–(b) show a unimodal sequence of W cookies and a unimodal sequence of E cookies, respectively.

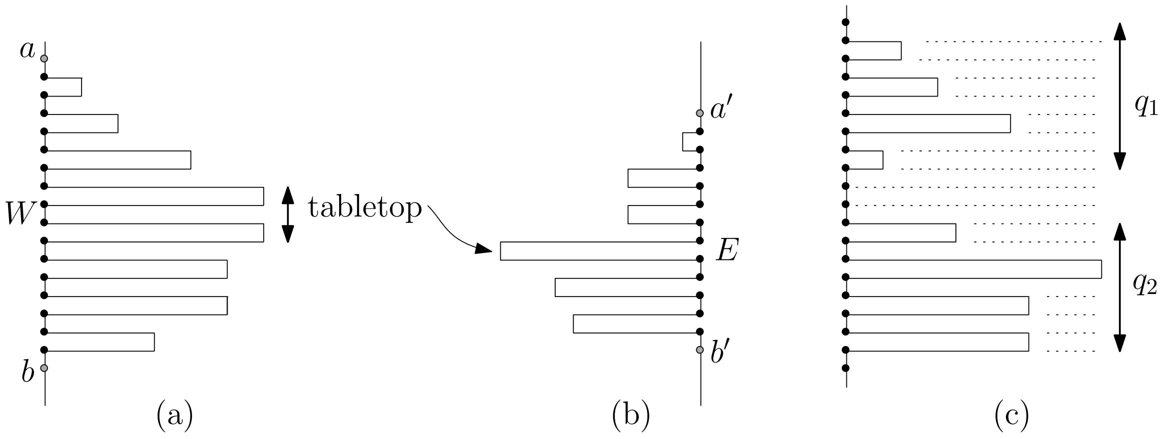


Figure 6.6: (a) A unimodal sequence of W cookies with two cookies in the tabletop, and (b) a unimodal sequence of E cookies with one cookie in the tabletop. (c) Two sequences q_1 and q_2 of W cookies in the subpath $P_{s,\alpha}$.

Note that according to our definition of *a set of cookies*, a set of cookies is a special type of unimodal sequence of cookies consisting of a tabletop that contains all the cookies in the set.

Let a and b be two vertices on the same boundary tp, where $tp \in \{N, S, E, W\}$ and $P_{s,t}$ visits a before b. The subpath $P_{a,b}$ is called an *extended sequence of tp cookies* if it contains either a sequence of tp cookies covering the vertices in the interval (a, b) on the tp boundary, or the vertices in the interval $[a = s, b)$, or the vertices in the interval $(a, b = t]$.

We establish the structure of the initial subpath of $P_{s,t}$ in the next theorem (refer to Figure 6.7), but before the theorem we make some remarks to prepare the reader for the numerous possible configurations of the initial subpath of an s, t Hamiltonian path even when it is only 1-complex.

Remark. Let P be a 1-complex s, t Hamiltonian path. Based on the existence of corner separators cutting off s in P and the structure and type of the first cross separators, the initial subpath of P can have so many different configurations that it is unwieldy to specify completely the structure of the initial subpath of P in the general case. In fact, we are about to define a special kind of 1-complex s, t Hamiltonian path with a precise structure that we call a "good Hamiltonian path" in Section 6.3.1 such that the initial subpath of such a path has a small number of configuration cases. Furthermore, we show by example in Section 6.3.2 that a 1-complex s, t Hamiltonian path can be reconfigured to a good Hamiltonian path. Thus, for purposes of obtaining a reconfigurartion result for 1-complex s, t Hamiltonian paths, we believe that the structure we are about to present is sufficiently detailed. As we will see, the concept of a good Hamiltonian path helps us avoid a lot of unnecessary case-handling.

Theorem 6.1 (Initial Subpath Structure)**.** *The initial subpath $P_{s,s(\eta_1)}$ consists of two subpaths $P_{s,\alpha}$ and $P_{\alpha,s(\eta_1)}$. The subpath $P_{s,\alpha}$ has the following structure.*

(a) If $P_{s,t}$ has no corner separators cutting off s then $P_{s,\alpha}$ contains either all the edges on the W boundary from s to α; or an extended unimodal sequence of W cookies $P_{s,\alpha}$; or an extended unimodal sequence of W cookies $P_{s,a}$, where a is a vertex on the W boundary and $0 < y(a) < y(\alpha)$, followed by all the edges on the W boundary from a to α; or all the edges on the W boundary from s to some vertex b on the boundary, where $0 < y(b) < y(\alpha)$, followed by an extended unimodal sequence of W cookies $P_{b,\alpha}$.

(b) If $P_{s,t}$ contains exactly one corner separator μ_1 cutting off s, then the subpath $P_{s,s(\mu_1)}$ is an extended sequence of N cookies with the size of all the N cookies being $y(b(\mu_1)) - 1$, followed by μ_1 such that $s(\mu_1)$ is on the N boundary and $t(\mu_1)$ is on the W boundary. The subpath $P_{t(\mu_1),\alpha}$ consists of either all the edges on the W boundary from $t(\mu_1)$ to α; or an extended unimodal sequence of W cookies $P_{t(\mu_1),\alpha}$; or an extended unimodal sequence of W cookies $P_{t(\mu_1),a}$, where a is some

vertex on the W boundary and $y(t(\mu_1)) < y(a) < y(\alpha)$, followed by all the edges on the W boundary from a to α.

(c) *If $P_{s,t}$ contains an odd number j of corner separators $\mu_1, \mu_2, \ldots, \mu_j$ cutting off s, where j is their total number, then the subpath $P_{s,s(\mu_1)}$ has the same form as the subpath $P_{s,s(\mu_1)}$ in Case (b), and the subpath $P_{t(\mu_j),\alpha}$ has the same form as the subpath $P_{t(\mu_1),\alpha}$ in Case (b). The endpoint $s(\mu_i)$, $1 \leq i \leq j$, is on the N boundary when i is odd and on the W boundary otherwise. If $t(\mu_i)$ and $s(\mu_{i+1})$, $1 \leq i < j$, are not adjacent in G then when i is odd $P_{t(\mu_i),s(\mu_{i+1})}$ is an extended sequence of W cookies such that all such cookies have size $x(b(\mu_i))$, and when i is even $P_{t(\mu_i),s(\mu_{i+1})}$ is an extended sequence of N cookies such that all such cookies have size $y(b(\mu_i))$.*

(d) *If $P_{s,t}$ contains an even number of corner separators $\mu_1, \mu_2, \ldots, \mu_j$, where $j > 1$ is even and equal to the total number of such separators, then the subpath $P_{s,s(\mu_1)}$ is an extended sequence of W cookies such that all such cookies have size $x(b(\mu_1))-1$. On $P_{s,\alpha}$, the subpath $P_{s,s(\mu_1)}$ is followed by μ_1 such that $s(\mu_1)$ is on the W boundary and $t(\mu_1)$ on the N boundary. The endpoint $s(\mu_i)$, $1 \leq i \leq j$, is on the N boundary when i is even and on the W boundary otherwise. If $t(\mu_i)$ and $s(\mu_{i+1})$, $1 \leq i < j$, are not adjacent in G then when i is even $P_{t(\mu_i),s(\mu_{i+1})}$ is an extended sequence of W cookies such that all such cookies have size $x(b(\mu_i))$, and when i is odd $P_{t(\mu_i),s(\mu_{i+1})}$ is an extended sequence of N cookies such that all such cookies have size $y(b(\mu_i))$. The subpath $P_{t(\mu_j),\alpha}$ has the same form as the subpath $P_{t(\mu_1),\alpha}$ in Case (b).*

The second subpath $P_{\alpha,s(\eta_1)}$ of $P_{s,s(\eta_1)}$ consists of either all the edges on the S boundary from α to $s(\eta_1)$; or an extended unimodal sequence of S cookies $P_{\alpha,s(\eta_1)}$; or all the edges on the S boundary from α to some vertex b on the boundary, where $0 < x(b) < x(s(\eta_1))$, and an extended unimodal sequence of S cookies $P_{b,s(\eta_1)}$.

Proof. We first prove the structure of the subpath $P_{s,\alpha}$.

(a) If $P_{s,t}$ has no corner separators cutting off s, then either $P_{s,\alpha}$ has some cookies or no cookies at all. If $P_{s,\alpha}$ has no cookies, then it must contain all the edges on the W boundary since it must cover all the vertices on that boundary by Observation 6.2.

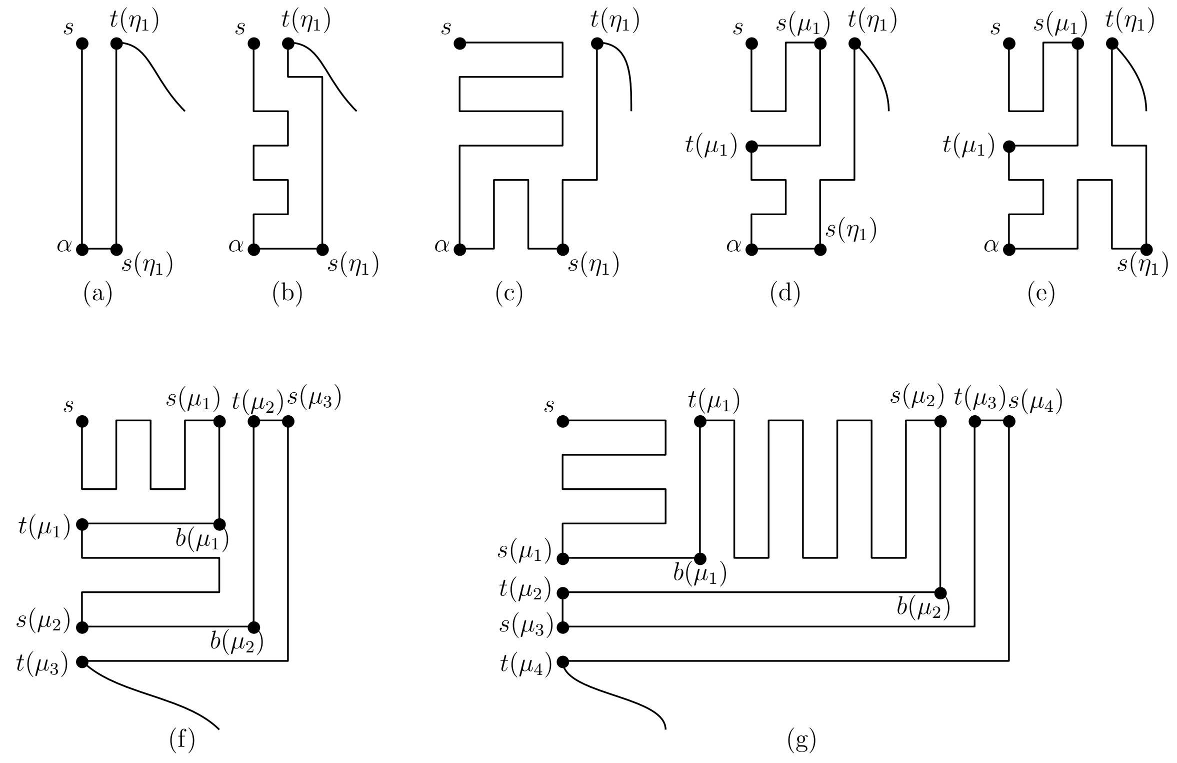


Figure 6.7: Examples of initial subpaths $P_{s,s(\eta_1)}$ of $P_{s,t}$.

If $P_{s,\alpha}$ has cookies, there cannot be any N cookies as there are no corner separators cutting off s. Suppose for a contradiction that there are S or E cookies in $P_{s,\alpha}$. Since all the E vertices must be covered by $P_{\beta,t}$ by Observation 6.2 and we assumed that $P_{s,t}$ visits α before β, subpath $P_{s,\alpha}$ cannot contain any E cookies. By Observation 6.3, there are no corner separators in $P_{s,t}$ that cut off α. Therefore, $P_{s,\alpha}$ cannot have any S cookies as well, a contradiction to our assumption. Hence, all the cookies in $P_{s,\alpha}$ must be from the W boundary.

Suppose for a contradiction that $P_{s,\alpha}$ contains a sequence of W cookies that has two tabletops as shown in Figure 6.6(c). By Lemma 6.1, $P_{s,t}$ contains at least one cross separator η_1 and it must have endpoints on the N and S boundaries. Then some vertices of Column 1 of G in the rows between the tabletops cannot be covered by any cookies since no E cookie can cross η_1 and no S cookies can cross the W cookies in the tabletop closer to the S boundary. Therefore any sequence of W cookies in $P_{s,\alpha}$ must be unimodal. By similar logic, we can prove that $P_{s,\alpha}$ cannot have more than one sequence of W cookies. The unimodal sequence of W cookies can extend from s to α; or can start at s but end at a vertex a on the W boundary before $P_{s,\alpha}$ reaches α, in which case subpath $P_{a,\alpha}$ consists of only edges on the W boundary from a to α; or can start at a vertex b on the W boundary below s and end at α. However, one of the endpoints of the extended sequence must be at either s or α; otherwise some vertices in Column 1 cannot be covered by $P_{s,t}$ without $P_{s,t}$ crossing itself. Hence $P_{s,t}$ fails to be both Hamiltonian and non-crossing, a contradiction.

(b) If there is exactly one corner separator μ_1 cutting off s, vertex $t(\mu_1)$ must be on the W boundary since $P_{s,\alpha}$ must cover all the W vertices by Observation 6.2. Then $s(\mu_1)$ must be on the N boundary since μ_1 cuts off s; see Figure 6.8(a). The subgrid G' of G with $s(\mu_1)$ and $t(\mu_1)$ as diagonally opposite corners must have size at least 3×3 as we do not consider a corner cookie as a separator. Then subpath $P_{s,s(\mu_1)}$ must be an extended sequence of N cookies and all those N cookies must have size $y(b(\mu_1)) - 1$.

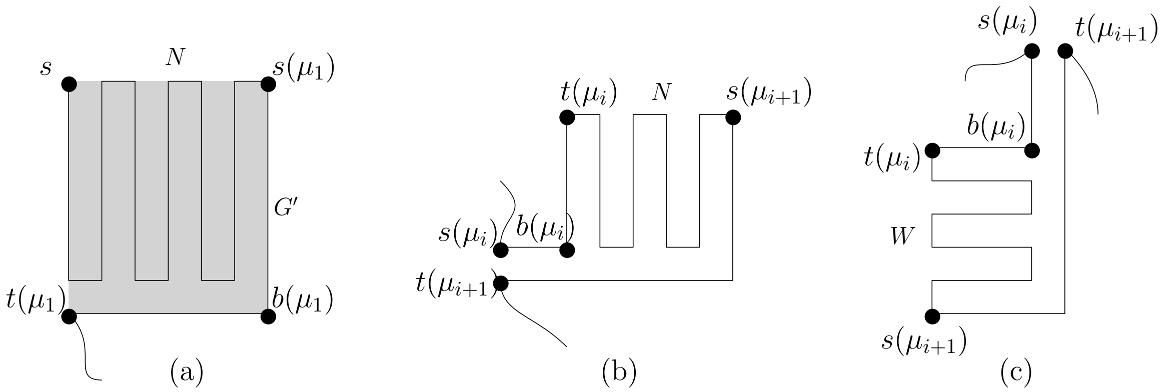


Figure 6.8: (a) The subpath $P_{s,t(\mu_1)}$. The subgrid G' that has $s(\mu_1)$ and $t(\mu_1)$ as diagonally opposite corners is shown in gray. (b) The vertices $s(\mu_i)$ and $t(\mu_{i+1})$ are on the W boundary, and $t(\mu_i)$ and $s(\mu_{i+1})$ are not adjacent on the N boundary. (c) The vertices $s(\mu_i)$ and $t(\mu_{i+1})$ are on the N boundary, and $t(\mu_i)$ and $s(\mu_{i+1})$ are not adjacent on the W boundary.

By logic similar to Case (a), we can prove that the subpath $P_{t(\mu_1),\alpha}$ contains either all the edges on the W boundary from $t(\mu_1)$ to α; or is an extended unimodal sequence of W cookies that must start at $t(\mu_1)$.

(c) If there are an odd number of corner separators $\mu_1, \mu_2, \ldots, \mu_j$ cutting off s, where j is the total number of such corner separators, the vertex $t(\mu_j)$ must be on the W boundary since $P_{s,\alpha}$ must cover all the W vertices by Observation 6.2. Then $s(\mu_j)$ must be on the N boundary. Then $s(\mu_{j-1})$ must be on the N boundary and

$t(\mu_{j-1})$ must be on the W boundary. Tracing $P_{t(\mu_j),s}$ this way, it is straightforward to see that $s(\mu_i)$, $1 \leq i \leq j$, lies on the N boundary when i is odd and $s(\mu_i)$ is on the W boundary when i is even. Therefore, $s(\mu_1)$ must be on the N boundary as in Case (b), and by logic similar to Case (b), the subpath $P_{s,s(\mu_1)}$ is an extended sequence of N cookies with the cookie size being $y(b(\mu_1))-1$ for all the N cookies.

The vertices $s(\mu_i)$ and $t(\mu_{i+1})$, $1 \leq i < j$, must be adjacent on the N or W boundary as otherwise some of the vertices on that boundary cannot be covered by $P_{s,t}$. First assume that i is even and $s(\mu_i)$ and $t(\mu_{i+1})$ are on the W boundary as shown in Figure 6.8(b). Then $t(\mu_i)$ and $s(\mu_{i+1})$ must be on the N boundary. If the vertices are adjacent in G, then there are no cookies between them. Otherwise subpath $P_{t(\mu_i),s(\mu_{i+1})}$ must be an extended sequence of N cookies and they must all have size $y(b(\mu_i))$.

Now assume that i is odd and $s(\mu_i)$ and $t(\mu_{i+1})$ are on the N boundary as shown in Figure 6.8(c). Then $t(\mu_i)$ and $s(\mu_{i+1})$ must be on the W boundary. If the vertices are not adjacent in G, then $P_{t(\mu_i),s(\mu_{i+1})}$ must be an extended sequence of W cookies and they must all have size $x(b(\mu_i))$.

Since $t(\mu_j)$ is on the W boundary, we can prove in a way similar to Case (a) that the subpath $P_{t(\mu_j),\alpha}$ either contains only edges on the W boundary; or is an extended unimodal sequence of W cookies and must start at $t(\mu_j)$.

(d) If there are an even number of corner separators $\mu_1, \mu_2, \ldots, \mu_j$ cutting off s, where j is the total number of such separators, the vertex $t(\mu_j)$ must be on the W boundary since $P_{s,\alpha}$ must cover all the W vertices by Observation 6.2. Then $s(\mu_j)$ must be on the N boundary. Then $s(\mu_{j-1})$ must be on the N boundary and $t(\mu_{j-1})$ must be on the W boundary. Tracing $P_{t(\mu_j),s}$ this way, it is straightforward to see that $s(\mu_i)$, $1 \leq i \leq j$, is on the N boundary when i is even and $s(\mu_i)$ is on the W boundary when i is odd. Therefore, $s(\mu_1)$ must be on the W boundary, and by logic similar to Case (b) the subpath $P_{s,s(\mu_1)}$ is an extended sequence of W cookies where the size of all the W cookies is $x(b(\mu_1)) - 1$.

The rest of the claims can be proved in a way similar to Case (c).

We now prove the statement following (d) which gives the structure of subpath $P_{\alpha,s(\eta_1)}$. Path $P_{s,\alpha}$ cannot reach any vertex on the S boundary other than α because there are no corner separators cutting off s by Observation 6.3, and no cross separators between the N and S boundaries before α by Lemma 6.1. Therefore, path $P_{\alpha,s(\eta_1)}$ must cover all the vertices on the S boundary from α to $s(\eta_1)$. Therefore, $P_{\alpha,s(\eta_1)}$ consists of either all the edges on the S boundary from α to $s(\eta_1)$; or an extended unimodal sequence of S cookies that must end at $s(\eta_1)$. □

We now establish the structure of the final subpath of $P_{s,t}$ before moving on to the middle subpath of $P_{s,t}$ (refer to Figure 6.9).

Theorem 6.2 (Final Subpath Structure)**.** *The final subpath $P_{t(\eta_k),t}$ consists of two subpaths $P_{t(\eta_k),\beta}$ and $P_{\beta,t}$.*

The subpath $P_{t(\eta_k),\beta}$ of $P_{s,t}$ consists of either all the edges on the N boundary from $t(\eta_k)$ to β; or an extended unimodal sequence of N cookies $P_{t(\eta_k),\beta}$; or an extended unimodal sequence of N cookies $P_{t(\eta_k),a}$, where a is some vertex on the N boundary and $x(t(\eta_k)) < x(a) < x(\beta)$, followed by all the edges on the N boundary from a to β; or all the edges on the N boundary from $t(\eta_k)$ to some vertex b on the boundary, where $x(t(\eta_k)) < x(b) < x(\beta)$, followed by an extended unimodal sequence of N cookies $P_{b,\beta}$.

The subpath $P_{\beta,t}$ has the following structure.

(a) If $P_{s,t}$ has no corner separators cutting off t then $P_{\beta,t}$ consists of either all the edges on the E boundary; or an extended unimodal sequence of E cookies $P_{\beta,t}$; or an extended unimodal sequence of E cookies $P_{\beta,a}$, where a is some vertex on the E boundary and $0 < y(a) < y(t)$, followed by all the edges on the E boundary from a to t; or all the edges on the E boundary from β to some vertex b on the boundary, where $0 < y(b) < y(t)$, followed by an extended unimodal sequence of E cookies $P_{b,t}$.

(b) If $P_{s,t}$ contains exactly one corner separator ν_1 cutting off t, then the subpath $P_{\beta,s(\nu_1)}$ consists of either all the edges on the E boundary from β to $s(\nu_1)$; or all the edges on the E boundary from β to vertex b on the boundary, where $0 < y(b) < y(t)$, followed by an extended unimodal sequence of E cookies $P_{b,s(\nu_1)}$. The subpath $P_{s(\nu_1),t}$ contains ν_1, with $s(\nu_1)$ on the E boundary and $t(\nu_1)$ on the S boundary, followed by an extended sequence of S cookies $P_{t(\nu_1),t}$ where the size of all such cookies must be $m - y(b(\nu_1)) - 1$.

(c) If $P_{s,t}$ contains more than one corner separator $\nu_1, \nu_2, \ldots, \nu_j$, where $j > 1$, then the subpath $P_{\beta,s(\nu_1)}$ is the same as the subpath $P_{\beta,s(\nu_1)}$ in Case (b). The endpoint $s(\nu_i)$, $1 \leq i \leq j$, is on the E boundary when i is odd and on the S boundary otherwise. If $t(\nu_i)$ and $s(\nu_{i+1})$, $1 \leq i < j$, are not adjacent in G then when i is

even, $P_{t(\nu_i),s(\nu_{i+1})}$ is an extended sequence of E cookies where the size of all such cookies is $n - x(b(\nu_{i+1}))$, and when i is odd, $P_{t(\nu_i),s(\nu_{i+1})}$ is an extended sequence of S cookies where the size of all such cookies is $m - y(b(\mu_{i+1}))$. If j is odd then the subpath $P_{t(\mu_j),t}$ is an extended sequence of S cookies where the size of all such cookies is $m - y(b(\mu_j)) - 1$; if i is even then $P_{t(\mu_j),t}$ is an extended sequence of E cookies where the size of all such cookies is $n - x(b(\mu_j)) - 1$.

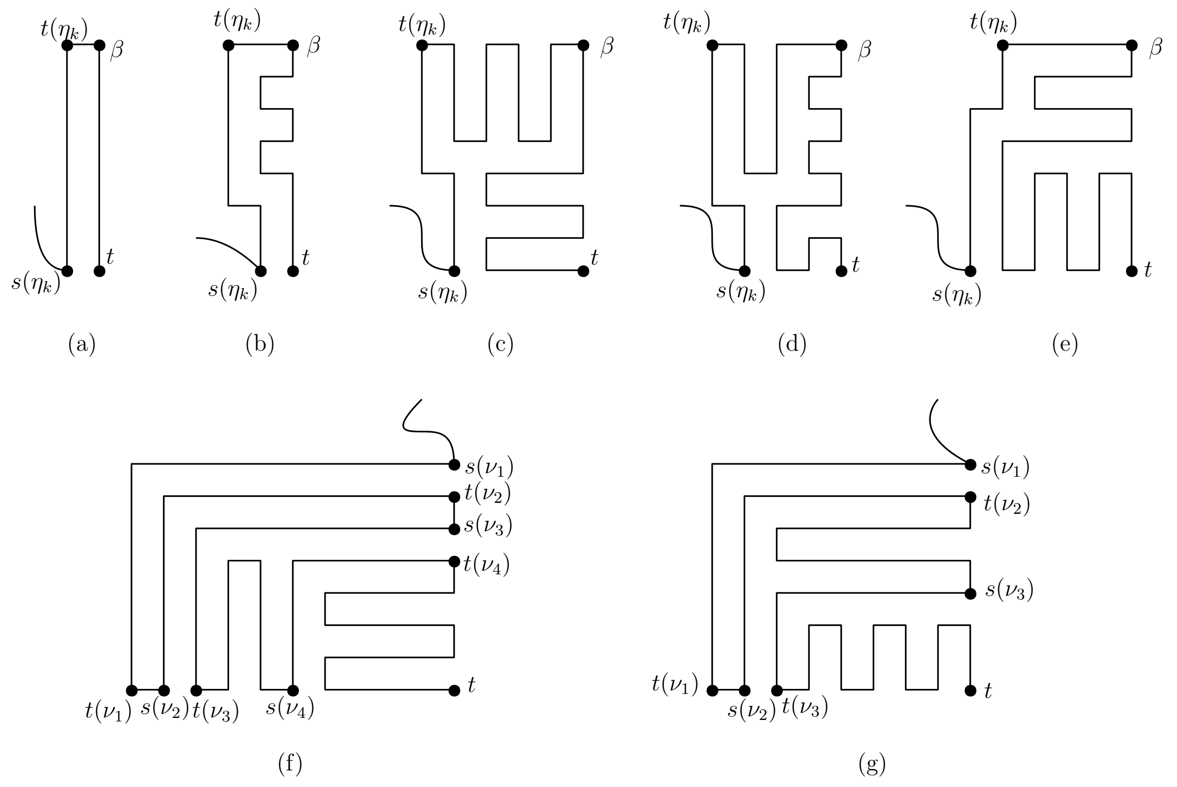


Figure 6.9: Examples of final subpaths $P_{t(\eta_k),t}$ of $P_{s,t}$.

Proof. The proof is similar to the proof of Theorem 6.1 after rotating the grid by 180 degrees and exchanging the roles of s and t. □

Figure 6.10 shows an example of the structure of the initial and final subpaths of $P_{s,t}$.

We have the following lemma which we use to establish the structure of the middle subpath of $P_{s,t}$.

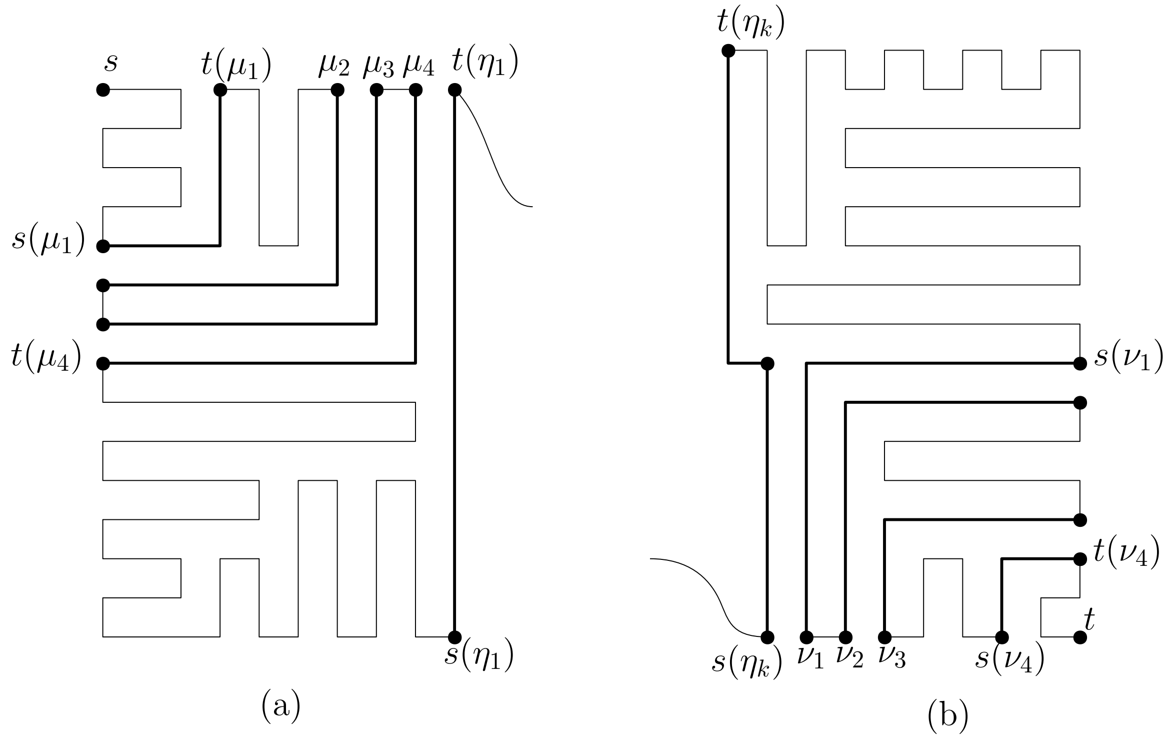


Figure 6.10: An example of (a) initial and (b) final subpaths of $P_{s,t}$. The corner separators cutting off s and t are shown in bold lines.

Lemma 6.2. *Let $P_{s,t}$ contain the cross separators $\eta_1, \eta_2, \ldots, \eta_k$, where $k > 1$. Then there are no cookies between η_p and η_{p+1}, where $1 \leq p < k$, if at least one of η_p and η_{p+1} is a straight separator (see Figure 6.11), and there is at most one cookie between η_p and η_{p+1} if both of them are bent separators.*

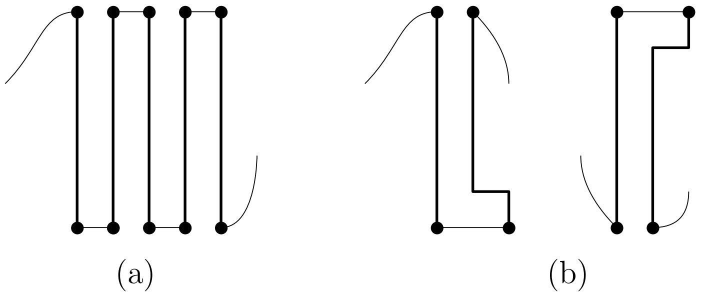


Figure 6.11: Two consecutive cross separators do not have cookies between them when (a) both are straight, or (b) one of them is straight and the other is bent.

Proof. Suppose for a contradiction that η_p and η_{p+1}, where $1 \leq p < k$, are both straight separators and that there is a cookie between them as shown in Figure 6.12(a).

Then $P_{s,t}$ fails to cover some vertices in the track the cookie lies in, and hence $P_{s,t}$ fails to be Hamiltonian. Therefore, η_p and η_{p+1} cannot have any cookies between them. Similarly we can prove that there cannot be any cookies between η_p and η_{p+1} if one of them is a straight separator and the other is a bent separator; see Figures 6.12(b)–(c).

Now assume that both η_p and η_{p+1} are bent separators and that they have more than one cookie between them in $P_{s,t}$, as shown in Figures 6.12(d)–(f). Then again $P_{s,t}$ fails to be Hamiltonian since some of the vertices in the vertical tracks where the cookies lie remain uncovered, a contradiction. □

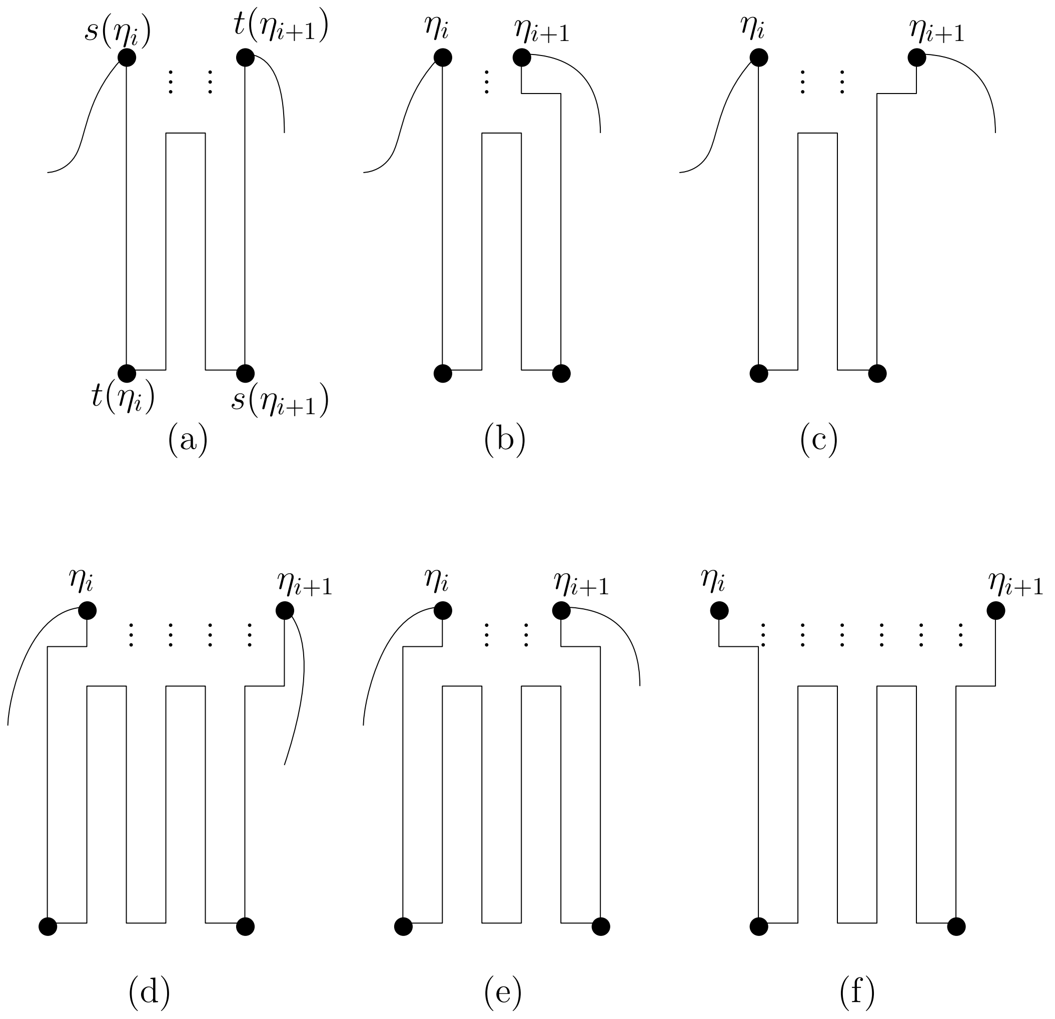


Figure 6.12: (a) A cookie between two straight separators, and (b)–(c) between a straight and a bent separator. (d)–(f) Two cookies between two bent separators.

We now establish the structure of the middle subpath $P_{s(\eta_1),t(\eta_k)}$ of $P_{s,t}$ in the following theorem. Refer to Figure 6.13.

Theorem 6.3 (Middle Subpath Structure)**.** *The structure of the middle subpath $P_{s(\eta_1),t(\eta_k)}$ of $P_{s,t}$ is as follows.*

(a) If $P_{s,t}$ contains just one cross separator η_1 then the middle subpath consists of the single cross separator.

(b) If $P_{s,t}$ contains the cross separators $\eta_1, \eta_2, \ldots, \eta_k$, where $k > 1$ is odd, then the cross separators occur as alternating groups of straight separators and groups of bent separators, starting with either type of group. A group of straight separators occupies consecutive columns. A group of bent separators $\eta_i, \ldots, \eta_j$, $1 \leq i < j \leq k$, has the following structure.

(i) If two bent separators η_p and η_{p+1}, where $i \leq p < j$, do not have any cookies between them, then either the internal bends of η_{p+1} are in the row below the internal bends of η_p, or the internal bends of η_{p+1} are in the row above the internal bends of η_p. (See Figures 6.13(b)–(c).)

(ii) If two bent separators η_p and η_{p+1}, where $i \leq p < j$, do have a cookie between them, then the internal bends of the bent separators are in the same row and there is an N or S cookie between η_p and η_{p+1}. (See Figure 6.13(a).)

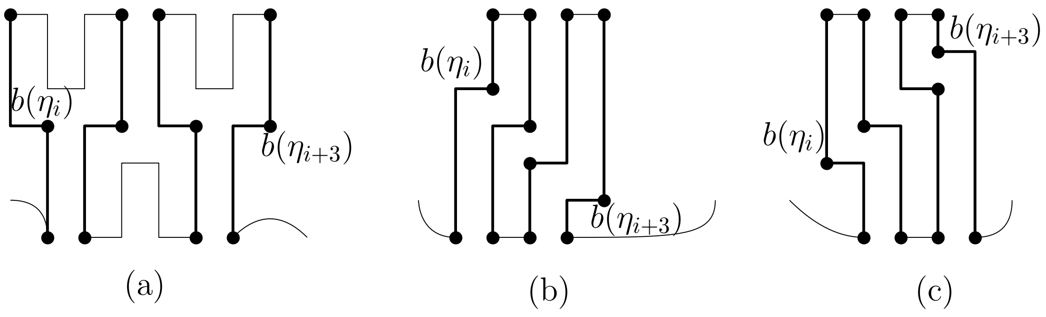


Figure 6.13: (a) A group of bent separators with their internal bends in the same row. (b) The $y(b(\eta_j))$'s of the group of the bent separators in descending order, and (c) in ascending order.

Proof. (a) The statement is clear from the definition of the middle subpath.

(b) By Lemma 6.1, $P_{s,t}$ must contain an odd number of cross separators. Since by Lemma 6.2, there cannot be any cookies between two consecutive straight separators, a group of straight separators must occupy consecutive columns.

(i) If the internal bends of η_p and η_{p+1}, where $i \leq p < j$, are not in the same row then they must occupy consecutive rows as otherwise some vertices in Rows $y(b(\eta_p)) + 1, y(b(\eta_p)) + 2, \ldots, y(b(\eta_{p+1})) - 1$ (when $y(b(\eta_p)) < y(b(\eta_{p+1})))$ or in Rows $y(b(\eta_p)) - 1, y(b(\eta_p)) - 2, \ldots, y(b(\eta_{p+1})) + 1$ (when $y(b(\eta_p)) > y(b(\eta_{p+1})))$ cannot be covered by $P_{s,t}$. By similar logic, η_p and η_{p+1} cannot have any cookies between them.

(ii) If the internal bends of η_p and η_{p+1}, where $i \leq p < j$, are in the same row then $x(s(\eta_{p+1}))$ must be three columns to the right of $x(t(\eta_p))$ and there must be an N or S cookie in the track t^v_x where $x = x(t(\eta_p)) + 1$.

□

6.3 Reconfiguration of s, t Hamiltonian Paths

In this section we give a research plan for obtaining reconfiguration results for any two 1-complex s, t Hamiltonian paths in a rectangular grid graph; parts of the plan are complete, while parts remain underway. Our plan builds on the structure of Hamiltonian paths we established in the previous sections, and makes use of the reconfiguration algorithms for cycles from Chapters 4 and 5.

Let P_1 and P_2 be two 1-complex s, t-Hamiltonian paths of G. To reconfigure P_1 to P_2, or vice versa, we apply a more elaborate approach than we did for reconfiguration of cycles. We still use canonical Hamiltonian paths as intermediate steps, but instead of reconfiguring P_1 (or P_2) directly to a canonical path, we reconfigure P_1 to a special 1-complex s, t Hamiltonian path P_1' (or P_2') that we call a "good Hamiltonian path" (defined later in this section) and then reconfigure P_1' (or P_2') to a canonical path as shown in Figure 6.14.

Our plan has the following steps. Step 1 is to reconfigure P_1 to a good Hamiltonian path P_1'. Step 2 is to reconfigure P_1' to a canonical Hamiltonian path $\mathcal{P}_1$. We then reconfigure P_2 to a (not necessarily different) canonical path $\mathcal{P}_2$ via another good Hamiltonian path P_2' in Steps 3 and 4. Step 5 is to reconfigure $\mathcal{P}_1$ to $\mathcal{P}_2$. Steps 6 and 7 are just reversing Steps 4 and 3, respectively, to reconfigure $\mathcal{P}_2$ to P_2.

Since Steps 1, 3 and 7 are similar, and Steps 2, 4 and 6 are similar, it suffices to give details only for Steps 1, 2 and 5.

The rest of this section is organized as follows. Section 6.3.1 defines a "good Hamiltonian path" which we later use as an intermediate step in our plan. Sections 6.3.2

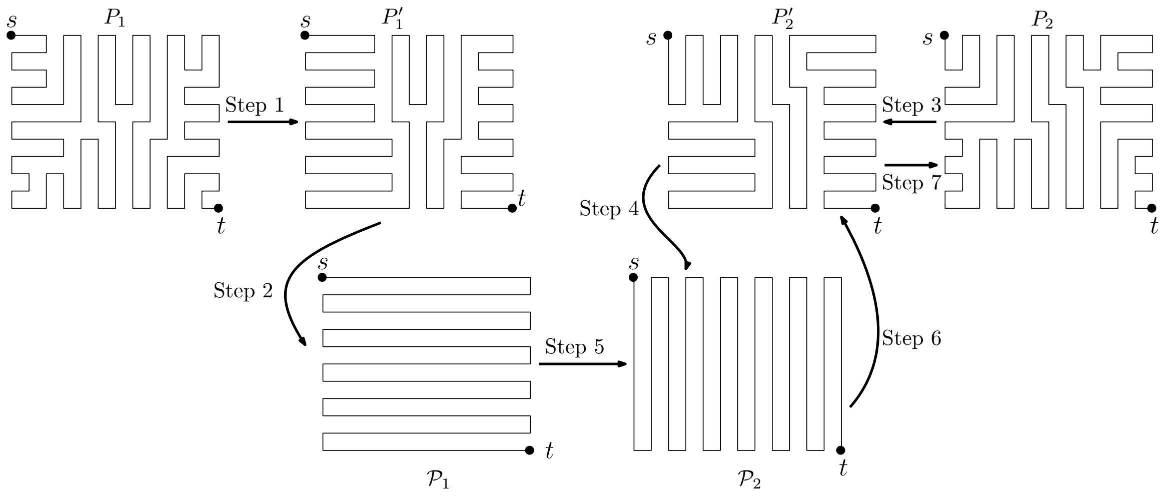


Figure 6.14: Reconfiguring P_1 to P_2, where both are 1-complex s, t Hamiltonian paths. The "good" paths P_1' and P_2' have simplified initial and final subpaths.

and 6.3.3 sketch outlines of algorithms to reconfigure any 1-complex path to a good Hamiltonian path (Step 1) and any good Hamiltonian path to a canonical path (Step 2), respectively. Note that these two steps are still underway and we do not provide a complete write-up of the algorithms and proofs of correctness. Section 6.3.4 gives an algorithm to reconfigure a canonical s, t Hamiltonian path to another canonical s, t Hamiltonian path, which is Step 5.

6.3.1 Good Hamiltonian Paths

In this section, we define a special 1-complex s, t Hamiltonian path we call a *good Hamiltonian path* which we use as an intermediate steps in our sketch of a reconfiguration algorithm.

Definition 6.6 (Good Initial Subpath). *The initial subpath of P, i.e., the subpath $P_{s,s(\eta_1)}$, is a* good initial subpath *if there is at most one corner separator cutting off s and the following two groups of conditions hold.*

(i) If there is a corner separator μ: η_1 is a bent separator, and there is exactly one set of N cookies between s and $s(\mu)$ of size $y(b(\mu)) - 1$ and at most one set of W cookies between $t(\mu)$ and $s(\eta_1)$ (Figure 6.15(h)).

(ii) If there is no corner separator: if η_1 is a straight separator then there is no cookies (Figure 6.15(a)) or one (Figure 6.15(b)) set of W cookies in the initial

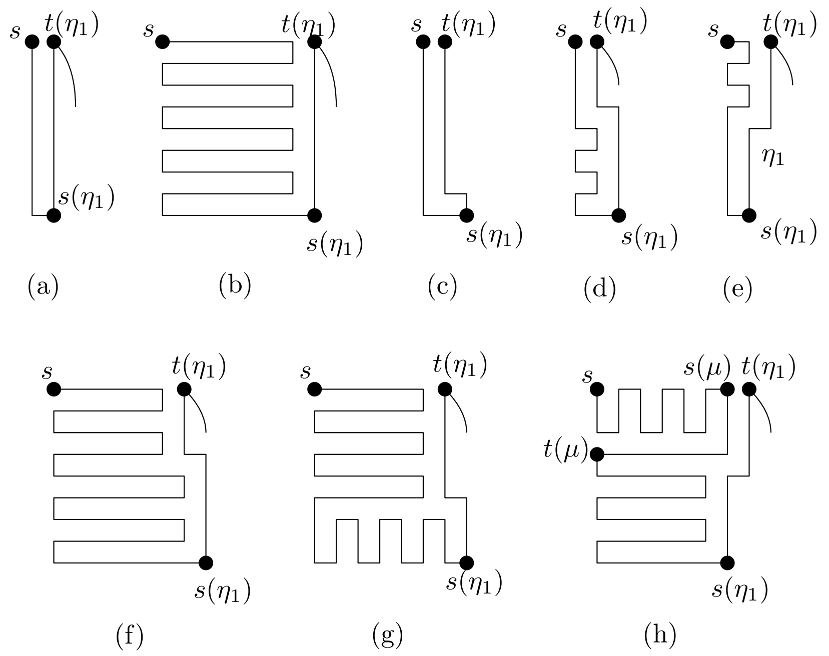


Figure 6.15: Good initial subpaths.

subpath, and if η_1 is a bent separator then η_1 is preceded by either boundary edges only (Figure 6.15(c)), or a set of W cookies (Figures 6.15(d)–(e)), or two sets of W cookies (Figure 6.15(f)), or a set of W cookies followed by a set of S cookies (Figure 6.15(g)).

A *good final subpath* is defined similarly, after interchanging the roles of s and t. We now define a good Hamiltonian path.

Definition 6.7 (Good Hamiltonian Path)**.** *A* 1*-complex s, t Hamiltonian path P is a* good Hamiltonian path *if both the initial and final subpaths are good.*

Observe that a *canonical Hamiltonian path $\mathcal{P}$* is a good Hamiltonian path that bends only on boundaries of G.

6.3.2 Step 1: Any 1-Complex to a Good Hamiltonian Path

This section gives examples of reconfiguring a 1-complex s, t Hamiltonian *path* to a good Hamiltonian path by applying algorithms for *cycles* from Chapters 4 and 5. We first show a simple example to illustrate the approach; we then present a more complicated case to give an idea about the difficulty of the problem.

Simple Example: No Corner Separators

Let P be a 1-complex s,t Hamiltonian path with no corner separators. To reconfigure P to a good Hamiltonian path, we have to transform the initial subpath of P to a good initial subpath and the final subpath of P to a good final subpath. As the two processes are similar if the roles of s and t are interchanged, we only discus how to reconfigure P to another 1-complex s,t Hamiltonian path P' such that P' has a good initial subpath.

Let G' be the subgrid of G induced by the subpath $P_{s,t(\eta_1)}$. First assume that η_1 is a straight separator and hence G' is rectangular as shown in Figure 6.16(a). We add another row above the N boundary of G' to obtain another rectangular grid graph G'^{+}. We then connect s to $t(\eta_1)$ through the new vertices of G'^{+}, creating a 1-complex Hamiltonian cycle C of G'^{+} with no E or N cookies. We then apply Algorithm RectThreeTypes from Chapter 4 to reconfigure C to a canonical cycle $\mathbb{C}$ that contains either a set of S cookies or a set of W cookies. We then remove the edges of $G'^{+} - G'$ to obtain a Hamiltonian path P' that has a good initial subpath. If $\mathbb{C}$ contains W cookies, then P' contains a set of W cookies in the initial subpath and the number of cross separators of P and P' are the same. If $\mathbb{C}$ contains S cookies, then P' contains more cross separators than P (see Figure 6.16(b)) and the initial subpath of P' contains only edges on the W and S boundaries and no cookies.

If η_1 is a bent separator as shown in Figure 6.16(c), then G' and G'^{+} are L-shaped grid graphs. We apply Algorithm XNEFE from Chapter 5 to reconfigure the cycle C' of G'^{+} to a canonical Hamiltonian cycle $\mathbb{C}'$ that contains no cookies from the east or north. We then remove the edges of $G'^{+} - G'$ to obtain a Hamiltonian path P'' that has a good initial subpath. P'' has the same number of cross separators as P. Based on the cookies of $\mathbb{C}'$, the initial subpath of P contains either two sets of W cookies (see Figure 6.16(d)), or one set of W cookies, or a set of W cookies and a set of S cookies.

A More Complicated Case

We now show an example of a case that is comparatively more complicated. Assume that P has an odd number j of corner separators $\mu_1, \mu_2, \ldots, \mu_j$ cutting off s and that η_1 is a bent separator as shown in Figure 6.17(a).

To obtain an s,t Hamiltonian path with a good initial subpath, we apply the following steps as shown in Figures 6.17(b)–(d).

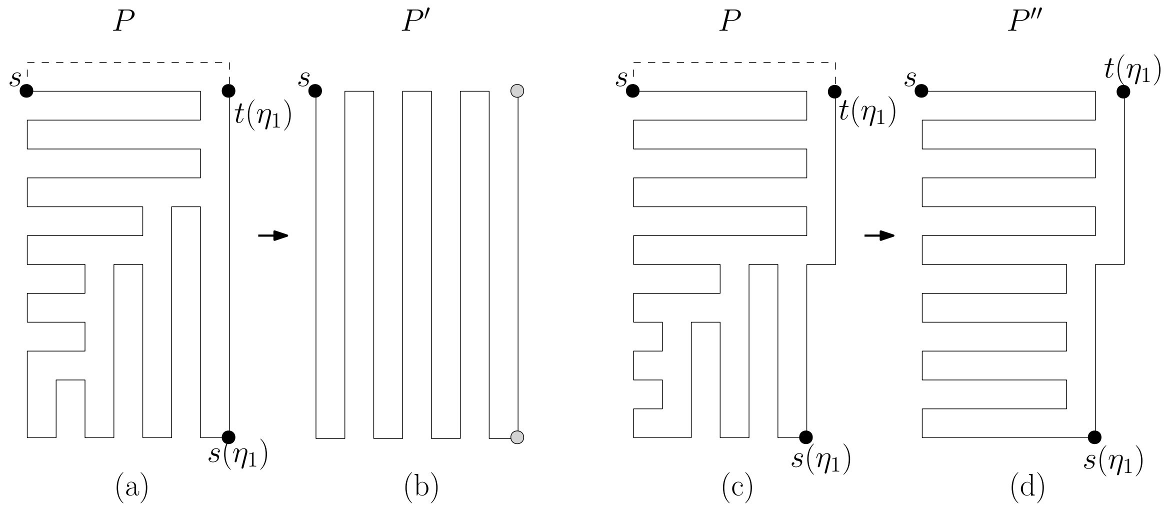


Figure 6.16: (a) Cross separator η_1 is a straight separator, s is connected to $t(\eta_1)$ by adding a dummy row above Row 0. (b) After applying Algorithm RectThreeTypes from Chapter 4 we have no cookies between s and $s(\eta_1)$ in P'. (c) The cross separator η_1 is a bent separator in P. (d) After applying Algorithm XNEFE from Chapter 5 we have two sets of W cookies between s and $s(\eta_1)$ in P''.

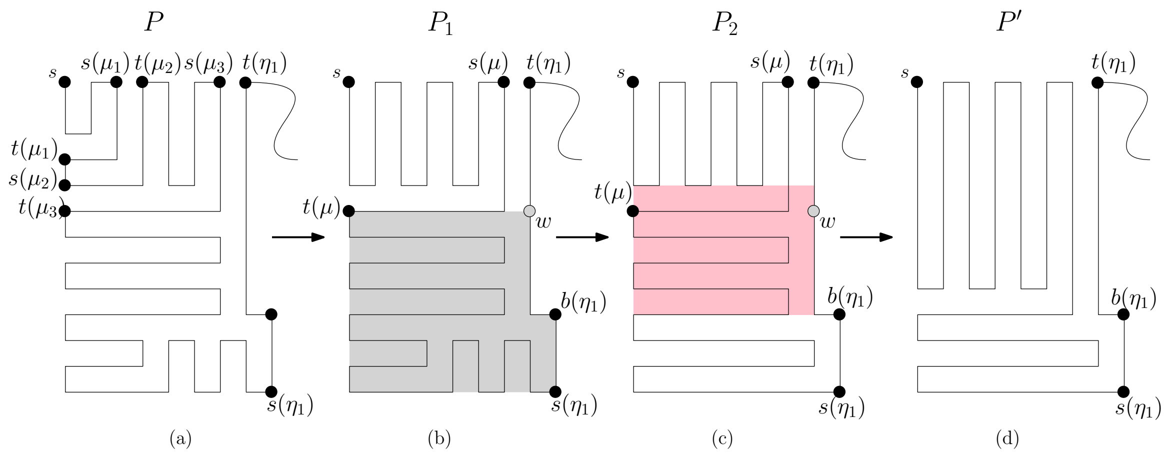


Figure 6.17: (a) P has an odd number j of corner separators $\mu_1, \mu_2, \ldots, \mu_j$ cutting off s and η_1 is a bent separator. (b) P_1 has just one corner separator μ in the initial subpath. (b) P_2 has two sets of cookies between $t(\mu)$ and $s(\eta_1)$. (d) P' has a good initial subpath.

1. Obtain P_1 from P such that P_1 has just one corner separator μ cutting off s, and the subpath $P_{t(\mu),t}$ of P_1 is the same as the subpath $P_{t(\mu_3),t}$ of P.

2. Take the subpath $P_{t(\mu),w}$ of P_1, where $w = v_{x(t(\eta_1)),y(t(\mu))}$. Connect $t(\mu)$ to w by

adding horizontal edges. Then apply the algorithm from Chapter 5 and remove the dummy edges to obtain a 1-complex s, t Hamiltonian path P_2 of G as shown in Figure 6.17(c).

3. If P_2 contains more than one set of cookies between the corner separator cutting off s and the first cross separator, apply transposes to grow the N cookies of P_2. We finally obtain P' such that P_2 has at most one set of cookies between the corner separator cutting off s and the first cross separator.

6.3.3 Step 2: Any Good to a Canonical Hamiltonian Path

In this section, we first show a simple example where the good Hamiltonian path contains a single cross separator, no corner separators, and has both E and W cookies. Our example illustrates the application of *switch* operations in pairs such that after applying each pair, we obtain another s, t Hamiltonian path. We call it a *base case* since it deals with paths with a single cross separator. We then give an example of a more *general case*, where the input path has more than one cross separator.

A Base Case

Let P be a good Hamiltonian path with no cross separators and a single cross separator η. For simplicity, we also assume that P has both E and W cookies. We show how to reconfigure P to a canonical Hamiltonian path using switch operations.

First assume that η is a straight separator as shown in Figure 6.18(a). We apply switch in the switchable cells that have an edge on η, working along η from $s(\eta)$ to $t(\eta)$; see Figure 6.18(b). Notice that the first switch operation produces a cycle-path cover of G containing a path from s to t and a cycle. The next switch operation *merges* the cycle and the path produced in the previous step to give an s, t Hamiltonian path of G. In this way every pair of switch operations produces an s, t Hamiltonian path of G. However, note that the intermediate Hamiltonian paths do not necessarily have bend complexity 1.

If η is a bent separator then again we apply switch in the switchable cells that have an edge on η, working along η from $s(\eta)$ to $t(\eta)$; see Figures 6.18(c)–(d).

Note that any pair of switch operations above that preserves Hamiltonicity is applied within a 4×3 or 3×4 subgrid.

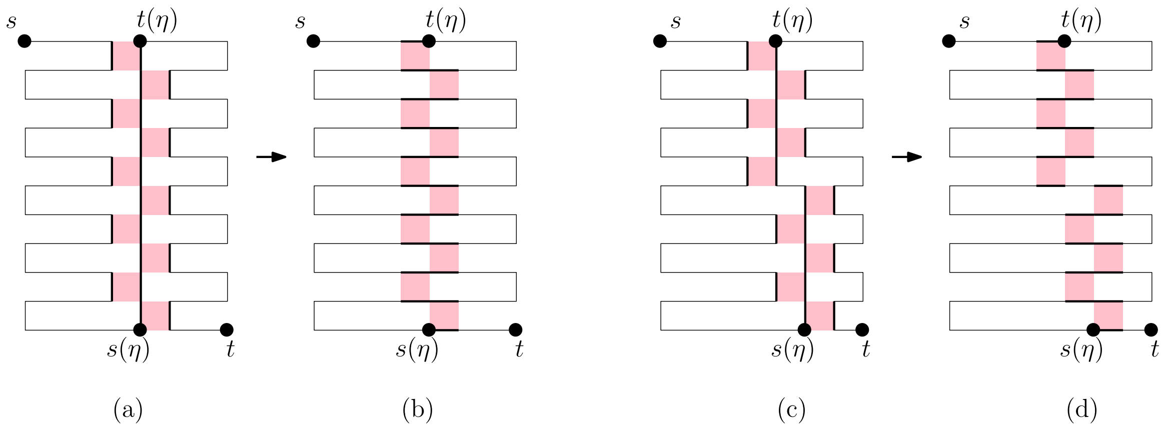


Figure 6.18: (a) P has only one cross separator η which is a straight separator, (b) after applying switches *along* η we have a canonical Hamiltonian path. (c) The only cross separator η is a bent separator, (d) the canonical Hamiltonian path after applying switches.

General Case

We now consider a good Hamiltonian path P such that P has $k = 2i + 1$ cross separators, where $i \geq 0$. To prove that P can be reconfigured to a canonical Hamiltonian path, we apply induction on i. The base case is $i = 0$, i.e., $k = 1$, as shown in the example above. Our induction hypothesis is that P can be reconfigured to a canonical path when $k = 2i + 1$, $i \geq 1$. We then show that our claim remains true for $k = 2(i + 1) + 1$ by obtaining a Hamiltonian path P' from P such that P' has $k' \leq k - 2$ cross separators.

Out of the many cases we have to consider based on the structure of the initial subpath of P, we present just one case here when the first three cross separators η_1, η_2, and η_3 of P are bent separators without any cookies between them, and there is a corner separator μ in P cutting off s.

We apply the following steps to obtain P' from P.

1. Obtain P_1 from P by applying two switches in the pink cells shown in Figure 6.19(b). Note that P_1 has bend complexity 2.

2. Apply transposes on P_1 to shorten the size of all the N cookies by 2 and obtain P_2. Note that P_2 is a 1-complex path. See Figure 6.19(c).

3. Take the subpath $P_{u,w}$ of P_2; connect u to w by a horizontal edge to obtain a rectangular 1-complex cycle C; apply the algorithm from Chapter 4 to make C

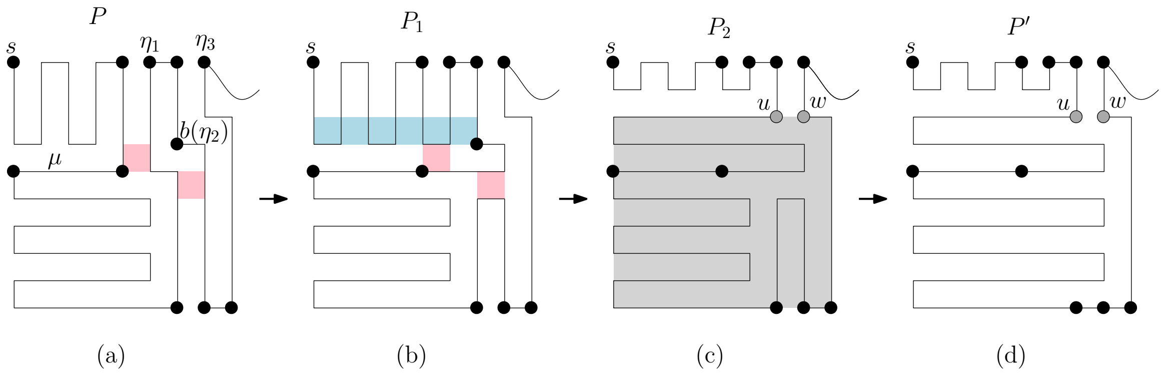


Figure 6.19: (a) The first three cross separators η_1, η_2, and η_3 of P are bent separators, and there is a corner separator μ in P cutting off s. (b) After applying switches in the pink cells, and (c) after applying transposes to shorten the N cookies. (d) P' has a good initial subpath.

canonical and then remove edge (u, w) to obtain P'. Path P' has a good initial subpath with a corner separator cutting off s, and at most one set of cookies between the corner separator cutting off s and the first cross separator.

6.3.4 Step 5: Canonical to Canonical Hamiltonian Path

In this section, we give an algorithm to reconfigure one canonical s, t Hamiltonian path to another such path. The algorithm uses algorithms from Chapter 4 to reconfigure between canonical Hamiltonian cycles in rectangular grid graphs.

Let G be an $m \times n$ grid graph, where m, n are odd, with the upper left corner s and lower right corner t. Recall from Chapter 2 that G can have two canonical s, t Hamiltonian paths when m and n are both odd. Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be the two canonical s, t Hamiltonian paths of G. Without loss of generality we assume that $\mathcal{P}_1$ bends at the E and W boundaries of G as shown in Figure 6.20(a) and that $\mathcal{P}_2$ bends at the N and S boundaries of G as shown in Figure 6.20(d). We give an algorithm CanToCanPath to reconfigure $\mathcal{P}_1$ to $\mathcal{P}_2$.

The main idea of this algorithm is as follows. To obtain a canonical cycle from $\mathcal{P}_1$, we first add a column to the right of the E boundary of G and a row above the N boundary of G to create an $(m + 1) \times (n + 1)$ grid graph G^+. We then connect s to t through vertices of $G^+ - G$ to obtain a canonical Hamiltonian cycle $\mathbb{C}_1$ of G^+ that consists of a set of W cookies and some boundary edges connecting the cookies. We apply the algorithm from Chapter 4 to reconfigure $\mathbb{C}_1$ to a canonical Hamiltonian

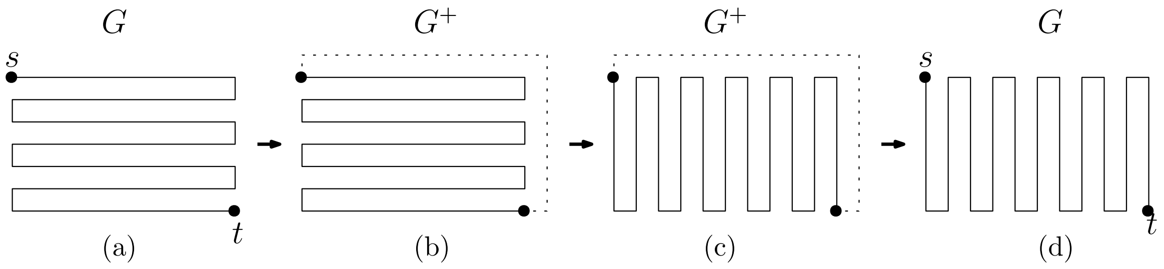


Figure 6.20: (a) Canonical Hamiltonian path $\mathcal{P}_1$ of G, (b) canonical Hamiltonian cycle $\mathbb{C}_1$ of G^+, (c) another canonical Hamiltonian cycle $\mathbb{C}_2$ of G^+, and (d) canonical Hamiltonian path $\mathcal{P}_2$ of G.

cycle $\mathbb{C}_2$ that contains S cookies. We then remove the edges of G^+ not in G to obtain canonical Hamiltonian path $\mathcal{P}_2$ of G. Figure 6.20 shows the steps of algorithm CanToCanPath.

We now prove the correctness and time complexity of the algorithm.

Theorem 6.4. *Algorithm* CanToCanPath *reconfigures $\mathcal{P}_1$ to $\mathcal{P}_2$ using $O(|G|)$ flip and transpose operations and runs in $O(|G|)$ time.*

Proof. It takes $O(|G|)$ time to create the canonical Hamiltonian cycle $\mathbb{C}_1$ from $\mathcal{P}_1$. The algorithm from Chapter 4 to reconfigure $\mathbb{C}_1$ to $\mathbb{C}_2$ takes $O(|G|)$ flips and transposes, and each such operation takes $O(1)$ time. Then it takes $O(|G|)$ time to remove the edges of G^+ that are not in G and thus to obtain $\mathcal{P}_2$ from $\mathbb{C}_2$. Therefore, Algorithm CanToCanPath requires $O(|G|)$ flip and transpose operations and runs in $O(|G|)$ time. □

6.4 Conclusion

In this chapter, we established the structure of a 1-complex s,t Hamiltonian path. We gave an algorithm to reconfigure one canonical Hamiltonian path to another canonical Hamiltonian path using reconfiguration algorithms from Chapters 4 and 5. We then sketched an idea for an algorithm to reconfigure any 1-complex s,t Hamiltonian path to any other such path in a rectangular grid graph using the structure we established in this chapter. Based on our research, we intend to complete the write-up of the details of the algorithm to reconfigure any 1-complex s,t Hamiltonian path P_1 to any other such path P_2 using $O(|G|^2)$ switch operations.

We conclude with some open problems.

1. Can we reconfigure P_1 to P_2 in $O(|G|)$ time?
2. Is it possible to reconfigure 1-complex Hamiltonian paths using only flips and transposes?
3. Can we reconfigure *cycle-path covers* (defined in Chapter 2) using the local operations *flip*, *transpose* and *switch*?

Chapter 7

Conclusion

In this chapter, we conclude our thesis by summarizing our findings presented in this thesis, and discussing some ongoing work and some open problems. In Section 7.1 we summarize our contributions and results. In Section 7.2 we discuss some ongoing work on generating 1-complex Hamiltonian cycles in rectangular grid graphs. Section 7.3 discusses some open problems and possible directions for future research.

7.1 Summary of Contributions

In this thesis, we studied reconfiguration of Hamiltonian paths and cycles in grid graphs. Our main contributions are listed below.

1. We studied the structure of Hamiltonian cycles and paths in grid graphs and introduced a complexity measure called *bend complexity* for Hamiltonian paths and cycles in grid graphs. Although we have studied only solid grid graphs in this thesis, the complexity measure is applicable to Hamiltonian cycles and paths in any grid graph.

2. We also measured complexity of a grid graph G based on the complexities of the Hamiltonian cycles and paths of G. We gave upper and lower bounds on the *bend complexity* of an $m \times n$ grid graph.

3. We defined three *local* operations, *flip*, *transpose* and *switch*, where *local* means that the operations are applied on vertices and edges that are close in the grid graph but may not be close on the path or cycle. Each of our operations is applied

in a subgrid of constant dimension in the *embedded grid graph* and therefore, computationally can be applied in constant time.

4. We showed that any Hamiltonian cycle can be reconfigured to any other Hamiltonian cycle in an $m \times n$ *rectangular* grid graph, where $m \leq 4$, using $O(|G|)$ flips and transposes, regardless of the bend complexities of the two cycles. We achieved similar results for s, t Hamiltonian paths. These results prove that the solution space graphs for the Hamiltonian cycle and path problems in narrow, i.e., when one of the dimensions is "small" (≤ 4), rectangular grids are connected and have $O(|G|)$ diameter.

5. We have given algorithms to reconfigure 1-complex Hamiltonian cycles in rectangular grid graphs using $O(|G|)$ flips and transposes, where the intermediate steps are also 1-complex Hamiltonian cycles. This result shows that the solution space graph for the 1-complex Hamiltonian *cycle* problem for *rectangular* grid graphs is connected under flip and transpose operations, and the solution space graph has $O(|G|)$ diameter. All our algorithms preserve the bend complexity of the cycle to 1 at each intermediate step.

6. We also showed that the solution space graph for the 1-complex Hamiltonian cycle problem for *L-shaped* grid graphs is connected under flip and transpose operations, and the solution space graph has $O(|G|)$ diameter, by giving algorithms where the cycle remains 1-complex at each intermediate step.

7. We established the structure of 1-complex Hamiltonian paths between diagonally opposite corners s and t of a rectangular grid graph. We gave an algorithm to reconfigure between canonical s, t Hamiltonian paths. We also provided a plan, based on work in progress, for designing an algorithm to reconfigure between any two 1-complex s, t Hamiltonian paths using *switch* operations.

7.2 Work in Progress: Generating 1-Complex Cycles

In addition to work in progress to complete the write-up of the reconfiguration algorithm for 1-complex s, t Hamiltonian paths in rectangular grid graphs, we have further work in progress. In this section, we give an overview of an algorithm to generate 1-complex Hamiltonian cycles in rectangular graphs with no E cookies. This algorithm could later form part of an algorithm to generate general 1-complex Hamiltonian cycles in rectangular grid graphs.

We first generate an initial cycle C' that has W cookies covering Rows 2 to $m-3$; see Figure 7.1 for the four different possibilities. Row 1 is covered by either a W cookie of maximum size (when W cookies occupy odd tracks) or N cookies of unit size (when W cookies occupy even tracks). Similarly, Row $m-2$ is covered by either a W cookie of maximum size or S cookies of unit size.

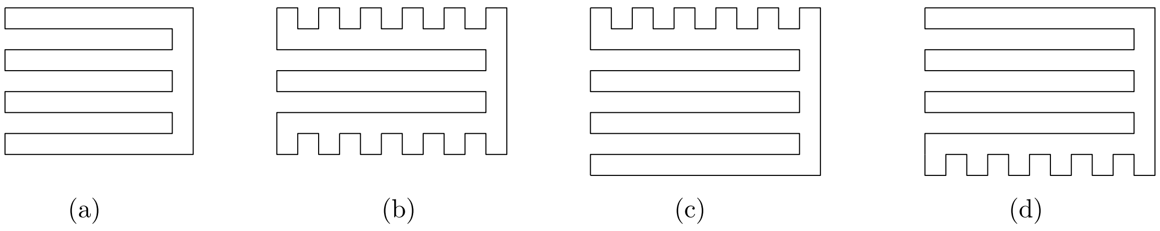


Figure 7.1: The initial cycles generated by the algorithm to generate Hamiltonian cycles with no E cookies: (a) m is even and the W cookies (if any) occupy odd tracks, (b) m is even and the W cookies occupy even tracks, (c) m is odd and the W cookies occupy even tracks, and (d) m is odd and the W cookies occupy odd tracks.

We then extend an N cookie in track t^v_{n-3}. We flip a coin and decide whether to apply a transpose: if “heads”, we apply a transpose on the N cookie. We can have at most $(m-2)/2$ coin flips and hence at most $(m-2)/2$ transposes this way if we get heads all the way. If we get “tails” at some point, we stop extending the N cookie and start extending an S cookie in the same track. Let the height of the N cookie in track t^v_{n-3} be y. Then we can have at most $(m-1-y)/2$ coin flips for the S cookie. If we get a tail at any point we stop extending the S cookie. If the N and S cookies are now facing each other in track t^v_{n-3}, we roll a three-sided die to decide whether to leave it like that, or to extend the N cookie using a flip, or to extend the S cookie by one row. We then move two tracks to the left. When we are at track t^v_{n-5}, we have to make sure that if there are some W cookies that go all the way to Column $n-2$, any N or S cookie in t^v_{n-5} cannot go beyond the tabletop of the sequence of W cookies. We carry on like this until we reach Column 0 (n is even) or Column 1 (n is odd).

Figure 7.2 shows some examples generated by our JAVA implementation of the above algorithm.

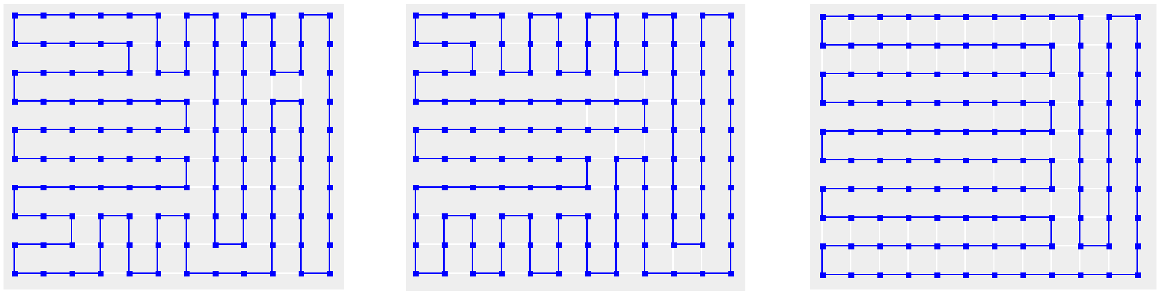


Figure 7.2: 1-complex cycles with no E cookies generated by our JAVA applet.

7.3 Open Problems

In this section we present some open problems and future directions of research.

1. When reconfiguring 1-complex cycles in rectangular grid graphs in Chapter 4, we applied the strategy of finding a *splitting track* and then splitting the reconfiguration problem into subproblems with restrictions on types of cookies. This approach was equally successful in reconfiguring Hamiltonian cycles in *L-shaped* grid graphs in Chapter 5. This leads us to the following conjecture.

   Conjecture. Let C be a 1-complex Hamiltonian cycle in a solid grid graph G with an arbitrary number of reflex corners. If C has cookies from all four directions (east, west, north, and south) then there exists a *splitting track* in C that has cookies of size at least one on both sides of the track.

2. Takaoka [116] showed that it is PSPACE complete to determine whether a Hamiltonian cycle can be reconfigured to any other Hamiltonian cycle using *switch* operations. Umans and Lenhart [119, 120], on the other hand, used a *switch* operation to "merge" cycles in a cycle cover in grid graphs into a single cycle, i.e., a Hamiltonian cycle. A natural extension of their work would be to study reconfiguration of cycle covers using *switch* operations.

   Open problem: Given two cycle covers of a grid graph G, can we reconfigure one to the other using only switch operations?

3. We can easily extend the definitions of flips and transposes to apply to cycle covers and to show that the application of flips and transposes to a cycle cover does not change the number of cycles in the set. A switch operation, however, may change

the number of cycles. Therefore we ask the following question.

Open problem: Given two cycle covers of a grid graph G where both the cycle covers contain the same number of cycles, can we reconfigure one to the other using only flip and transpose operations without changing the number of cycles in the cycle covers at the intermediate configurations?

4. Suppose a grid graph G has a Hamiltonian cycle. Does there exist a 1-complex Hamiltonian cycle in G? What is the computational complexity of finding one? What are the necessary and sufficient conditions?

5. Can we find an arbitrary solid grid graph with an even number of vertices that does not have a Hamiltonian cycle?

Bibliography

[1] Foto Afrati. The hamilton circuit problem on grids. *RAIRO - Theoretical Informatics and Applications - Informatique Theorique et Applications*, 28(6): 567–582, 1994.

[2] Alok Aggarwal, Don Coppersmith, Sanjeev Khanna, Rajeev Motwani, and Baruch Schieber. The angular-metric traveling salesman problem. *SIAM Journal on Computing*, 29(3):697–711, 2000.

[3] Soroush Alamdari, Patrizio Angelini, Fidel Barrera-Cruz, Timothy M Chan, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Penny Haxell, Anna Lubiw, Maurizio Patrignani, et al. How to morph planar graph drawings. *SIAM Journal on Computing*, 46(2):824–852, 2017.

[4] David L. Applegate, Robert E. Bixby, Vašek Chvátal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study.* Princeton University Press, 2006.

[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 S. Rao Kosaraju, editor, *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), Washington, D.C., USA*, pages 138–147, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics.

[6] Esther M. Arkin, Sándor P. Fekete, Kamrul Islam, Henk Meijer, Joseph S.B. Mitchell, Yurai Núñez Rodríguez, Valentin Polishchuk, David Rappaport, and Henry Xiao. Not being (super)thin or solid is hard: A study of grid hamiltonicity. *Computational Geometry*, 42(6-7):582–605, 2009.

[7] Sanjeev Arora. Polynomial time approximation schemes for euclidean traveling

salesman and other geometric problems. *The Journal of the ACM*, 45(5):753—-782, 1998.

[8] W. W. R. Ball and H. S. M. Coxeter. *Mathematical Recreations and Essays.* University of Toronto Press, Toronto, twelfth edition, 1974.

[9] J. C. Bermond and C. Thomassen. Cycles in digraphs – a survey. *Journal of Graph Theory*, 5(1):1–43, 1981.

[10] N. Biggs, E. K. Lloyd, and R. J. Wilson. *Graph Theory, 1736-1936.* Clarendon Press, USA, 1986.

[11] Markus Bläser. A new approximation algorithm for the asymmetric tsp with triangle inequality. *ACM Transactions on Algorithms*, 4(4), 2008.

[12] Olga Bodroža-Pantić, Bojana Pantić, Ilija Pantić, and Marija Bodroža-Solarov. Enumeration of hamiltonian cycles in some grid graphs. *MATCH - Communications in Mathematical and in Computer Chemistry*, 70:181–204, 01 2013.

[13] Béla Bollobás. *Modern Graph Theory.* Graduate texts in mathematics. Springer, Heidelberg, 1998.

[14] Marthe Bonamy, Matthew Johnson, Ioannis Lignos, Viresh Patel, and Daniël Paulusma. Reconfiguration graphs for vertex colourings of chordal and chordal bipartite graphs. *Journal of Combinatorial Optimization*, 27(1):132–143, 2014.

[15] J. A. Bondy. Properties of graphs with constraints on degrees. *Studia Scientiarum Mathematicarum Hungarica*, 4:473–475, 1969.

[16] Paul Bonsma. The complexity of rerouting shortest paths. *Theoretical Computer Science*, 510:1–12, 2013.

[17] Paul Bonsma and Luis Cereceda. Finding paths between graph colourings: Pspace-completeness and superpolynomial distances. *Theoretical Computer Science*, 410(50):5215–5226, 2009. Mathematical Foundations of Computer Science (MFCS 2007).

[18] Prosenjit Bose. Flips. In Walter Didimo and Maurizio Patrignani, editors, *Proceedings of the 20th International Symposium on Graph Drawing (GD), Redmond, WA, USA*, volume 7704 of *Lecture Notes in Computer Science*, page 1. Springer, 2012.

[19] M. Bousquet-Mélou, A. J. Guttmann, and I. Jensen. Self-avoiding walks crossing a square. *Journal of Physics A: Mathematical and General*, 38(42):9159, 2005.

[20] S. Bridgeman. Finding Hamiltonian Cycles in Grid Graphs without Holes. Bachelor of Arts with honors thesis, Williams College, Williamstown, Massachusetts, USA, 1995.

[21] H. J. Broersma. On some intriguing problems in hamiltonian graph theory — a survey. *Discrete Mathematics*, 251(1):47 – 69, 2002. Cycles and Colourings.

[22] Robert Cannon and Stan Dolan. The knight's tour. *Mathematics Magazine*, 70 (452):91–100, 1986.

[23] Luis Cereceda, Jan Van Den Heuvel, and Matthew Johnson. Connectedness of the graph of vertex-colourings. *Discrete Mathematics*, 308(5-6):913–919, 2008.

[24] Luis Cereceda, Jan Van Den Heuvel, and Matthew Johnson. Finding paths between 3-colorings. *Journal of Graph Theory*, 67(1):69–82, 2011.

[25] Shao Dong Chen, Hong Shen, and Rodney Topor. An efficient algorithm for constructing hamiltonian paths in meshes. *Parallel Computing*, 28(9):1293–1305, 2002.

[26] Hwan-Gue Cho and Alexander Zelikovsky. Spanning closed trail and hamiltonian cycle in grid graphs. In John Staples, Peter Eades, Naoki Katoh, and Alistair Moffat, editors, *Proceedings of the sixth International Symposium on Algorithms and Computations (ISAAC), Cairns, Australia*, volume 1004 of *Lecture Notes in Computer Science*, pages 342–351, Berlin, Heidelberg, 1995. Springer.

[27] Vašek Chvátal. On hamilton's ideals. *Journal of Combinatorial Theory*, 12: 163–168, 1972.

[28] Vašek Chvátal and Paul Erdős. A note on hamiltonian circuits. *Discrete Mathematics*, 2:111–113, 1972.

[29] Karen L. Collins and Lucia B. Krompart. The number of hamiltonian paths in a rectangular grid. *Discrete Mathematics*, 169(1–3):29–38, 1997.

[30] Axel Conrad, Tanja Hindrichs, Hussein Morsy, and Ingo Wegener. Solution of the knight's hamiltonian path problem on chessboards. *Discrete Applied Mathematics*, 50(2):125 – 134, 1994.

[31] A. R. Conway, I. G. Enting, and A. J. Guttmann. Algebraic techniques for enumerating self-avoiding walks on the square lattice. *Journal of Physics A: Mathematical and General*, 26(7):1519, 1993.

[32] G. A. Croes. A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812, 1958.

[33] Stephen J. Curran and Joseph A. Gallian. Hamiltonian cycles and paths in cayley graphs and digraphs — a survey. *Discrete Mathematics*, 156(1):1 – 18, 1996.

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

[35] Mark de Berg, Bart M. P. Jansen, and Debankur Mukherjee. Independent-set reconfiguration thresholds of hereditary graph classes. *Discrete Applied Mathematics*, 250:165–182, 2018.

[36] Erik D. Demaine, Martin L. Demaine, Eli Fox-Epstein, Duc A. Hoang, Takehiro Ito, Hirotaka Ono, Yota Otachi, Ryuhei Uehara, and Takeshi Yamada. Linear-time algorithm for sliding tokens on trees. *Theoretical Computer Science*, 600: 132–142, 2015.

[37] J. des Cloizeaux and G. Jannik. *Polymers in solution: their modelling and structure*. Clarendon Press, Oxford, 1987.

[38] Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, Heidelberg, New York, fourth edition, 2010.

[39] G. A. Dirac. Some theorems on abstract graphs. *Proceedings of the London Mathematical Society*, s3-2(1):69–81, 1952.

[40] A. P. Domoryad. *Mathematical Games and Pastimes*. Pergamon Press, Oxford, 1964.

[41] H. E. Dudeney. *Amusements in Mathematics*. Thomas, Springfield, 1917.

[42] H. E. Dudeney. *Amusements in Mathematics.* Dover, New York, 1970.

[43] Leonhard Euler. Solution d'une question curieuse qui ne paroit soumise à aucune analyse. *Mem. Acad. Sci. Berlin*, pages 310–337, 1759.

[44] Hazel Everett. Hamiltonian paths in nonrectangular grid graphs. Master's thesis, University of Saskatchewan, Canada, 1986.

[45] B. R. Feiring. An efficient procedure for obtaining feasible solutions to the n-city traveling salesman problem. *Mathematical and Computer Modelling*, 13(3): 67 – 71, 1990.

[46] 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 Dimitrios M. Thilikos, editor, *Proceedings of the 36th International Workshop on Graph Theoretic Concepts in Computer Science (WG), Zarós, Crete, Greece*, volume 6410 of *Lecture Notes in Computer Science*, pages 123–134, 2010.

[47] M. M. Flood. The traveling-salesman problem. *Operations Research*, 4(1): 61–75, 1956.

[48] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences).* W. H. Freeman, first edition, 1979.

[49] Michael R. Garey, R. L. Graham, and David S. Johnson. Some np-complete geometric problems. In Ashok K. Chandra, Detlef Wotschke, Emily P. Friedman, and Michael A. Harrison, editors, *Proceedings of the Eighth Annual ACM Symposium on Theory of Computing (STOC), Hershey, Pennsylvania, USA*, pages 10–22, New York, NY, USA, 1976. Association for Computing Machinery.

[50] Michael R. Garey, David S. Johnson, and R. E. Tarjan. The planar hamiltonian circuit problem is np-complete. *SIAM Journal on Computing*, 5(4):704–714, 1976.

[51] F. Göbel. On the number of hamilton cycles in product graphs. *Technische Hogeschool Twente*, 1979.

[52] Parikshit Gopalan, Phokion G. Kolaitis, Elitza Maneva, and Christos H. Papadimitriou. The connectivity of boolean satisfiability: Computational and structural dichotomies. *SIAM Journal on Computing*, 38(6):2330–2355, 2009.

[53] Anna Gorbenko and Vladimir Popov. On hamilton paths in grid graphs. *Advanced Studies in Theoretical Physics*, 7(3):127–130, 2013.

[54] Anna Gorbenko, Vladimir Popov, and Andrey Sheka. Localization on discrete grid graphs. In Xingui He, Ertian Hua, Yun Lin, and Xiaozhu Liu, editors, *Proceedings of the International Conference on Computer, Informatics, Cybernetics and Applications (CICA), Hangzhou, China*, volume 107 of *Lecture Notes on Electrical Engineering*, pages 971–978, Dordrecht, 2011. Springer Netherlands.

[55] Valery S. Gordon, Yury L. Orlovich, and Frank Werner. Hamiltonian properties of triangular grid graphs. *Discrete Mathematics*, 308(24):6166–6188, 2008.

[56] Ronald J. Gould. Updating the hamiltonian problem — a survey. *Journal of Graph Theory*, 15(2):121–157, 1991.

[57] Arash Haddadan, Takehiro Ito, Amer E. Mouawad, Naomi Nishimura, Hirotaka Ono, Akira Suzuki, and Youcef Tebbal. The complexity of dominating set reconfiguration. *Theoretical Computer Science*, 651:37–49, 2016.

[58] J. H. Halton and R. Terada. A fast algorithm for the euclidean traveling salesman problem, optimal with probability one. *SIAM Journal on Computing*, 11 (1):28–46, 1982.

[59] William Rowan Hamilton. Account of the icosian calculus. In *Royal Irish Academy*, volume 6, pages 415–416, 1858.

[60] Tatsuhiko Hatanaka, Takehiro Ito, and Xiao Zhou. The list coloring reconfiguration problem for bounded pathwidth graphs. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, 98(6): 1168–1178, 2015.

[61] Robert A. Hearn and Erik D. Demaine. Pspace-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1-2):72–96, 2005.

[62] Martin Held. *On the Computational Geometry of Pocket Machining*. Springer-Verlag, Berlin, Heidelberg, 1991.

[63] Michael Held and Richard M. Karp. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics*, 10(1):196–210, 1962.

[64] Kaiying Hou and Jayson Lynch. The computational complexity of finding hamiltonian cycles in grid graphs of semiregular tessellations. In Stephane Durocher and Shahin Kamali, editors, *Proceedings of the 30th Canadian Conference on Computational Geometry (CCCG), University of Manitoba, Winnipeg, Manitoba, Canada*, pages 114–128, 2018.

[65] Katherine Humphreys. A history and a survey of lattice path enumeration. *Journal of Statistical Planning and Inference*, 140(8):2237 – 2254, 2010. Lattice Path Combinatorics and Applications.

[66] S. P. Hurd and D. A. Trautman. The knight's tour on the 15-puzzle. *Mathematics Magazine*, 66(3):159–166, 1993.

[67] Kamrul Islam, Henk Meijer, Yurai Núñez Rodríguez, David Rappaport, and Henry Xiao. Hamilton circuits in hexagonal grid graphs. In Prosenjit Bose, editor, *Proceedings of the 19th Annual Canadian Conference on Computational Geometry (CCCG), Carleton University, Ottawa, Canada*, pages 85–88. Carleton University, Ottawa, Canada, 2007.

[68] Alon Itai, Christos H. Papadimitriou, and Jayme Luiz Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, 1982.

[69] Takehiro Ito and Erik D. Demaine. Approximability of the subset sum reconfiguration problem. *Journal of Combinatorial Optimization*, 28(3):639–654, 2014.

[70] Takehiro Ito, Erik D. Demaine, Nicholas J.A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. *Theoretical Computer Science*, 412(12):1054–1065, 2011.

[71] Takehiro Ito, Marcin Kamiński, and Erik D. Demaine. Reconfiguration of list edge-colorings in a graph. *Discrete Applied Mathematics*, 160(15):2199–2207, 2012.

[72] Takehiro Ito, Kazuto Kawamura, Hirotaka Ono, and Xiao Zhou. Reconfiguration of list l (2, 1)-labelings in a graph. *Theoretical Computer Science*, 544: 84–97, 2014.

[73] Takehiro Ito, Hirotaka Ono, and Yota Otachi. Reconfiguration of cliques in a graph. In Rahul Jain, Sanjay Jain, and Frank Stephan, editors, *Proceedings of the Twelfth Annual Conference on Theory and Applications of Models of Computation (TAMC), Singapore*, volume 9076 of *Lecture Notes in Computer Science*, pages 212–223. Springer, 2015.

[74] Takehiro Ito, Hiroyuki Nooka, and Xiao Zhou. Reconfiguration of vertex covers in a graph. *IEICE Transactions on Information and Systems*, 99(3):598–606, 2016.

[75] Takehiro Ito, Naonori Kakimura, Naoyuki Kamiyama, Yusuke Kobayashi, and Yoshio Okamoto. Reconfiguration of maximum-weight b-matchings in a graph. *Journal of Combinatorial Optimization*, 37(2):454–464, 2019.

[76] Hiroaki Iwashita, Yoshio Nakazawa, Jun Kawahara, Takeaki Uno, and Shin ichi Minato. Efficient computation of the number of paths in a grid graph with minimal perfect hash functions. Technical Report TCS-TR-A-13-64, Division of Computer Science, Hokkaido University, April 2013.

[77] J. L. Jacobsen. Exact enumeration of hamiltonian circuits, walks and chains in two and three dimensions. *Journal of Physics A: Mathematical and General*, 40:14667–14678, 2007.

[78] Marcin Kamiński, Paul Medvedev, and Martin Milanič. Complexity of independent set reconfigurability problems. *Theoretical Computer Science*, 439:9–15, 2012.

[79] Richard Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[80] Fatemeh Keshavarz-Kohjerdi and Alireza Bagheri. Hamiltonian paths in l-shaped grid graphs. *Theoretical Computer Science*, 621:37–56, 2016.

[81] Donald E. Knuth. *The Art of Computer Programming, Volume 4, Fascicle 1: Bitwise Tricks & Techniques; Binary Decision Diagrams*. Addison-Wesley Professional, Boston, MA, USA, 2009.

[82] Daniela Kühn and Deryk Osthus. A survey on hamilton cycles in directed graphs. *European Journal of Combinatorics*, 33(5):750 – 766, 2012. EuroComb '09.

[83] Y. H. Harris Kwong. Enumeration of hamiltonian cycles in $p_4 \times p_n$ and $p_5 \times p_n$. *Ars Combinatoria*, 33:87–96, 1992.

[84] Y. H. Harris Kwong and D. G. Rogers. A matrix method for counting hamiltonian cycles on grid graphs. *European Journal of Combinatorics*, 15(3):277—-283, 1994.

[85] Gilbert Laporte. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2):231 – 247, 1992.

[86] E. L. Lawler. *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley-Interscience series in discrete mathematics and optimization. John Wiley & Sons, 1985.

[87] Hao Li. Generalizations of dirac's theorem in hamiltonian graph theory — a survey. *Discrete Mathematics*, 313(19):2034 – 2053, 2013. Cycles and Colourings 2011.

[88] Ioannis Lignos. *Reconfigurations of Combinatorial Problems: Graph Colouring and Hamiltonian Cycle*. PhD thesis, Durham University, 2017.

[89] Shun-Shii Lin and Chung-Liang Wei. Optimal algorithms for constructing knight's tours on arbitrary $n \times m$ chessboards. *Discrete Applied Mathematics*, 146(3):219 – 232, 2005.

[90] John D. C. Little, Katta G. Murty, Dura W. Sweeney, and Caroline Karel. An algorithm for the traveling salesman problem. *Operational Research*, 11(6): 972–989, 1963.

[91] F. Luccio and C. Mugnia. Hamiltonian paths on a rectangular chessboard. In M. B. Pursley and J. B. Cruz, editors, *Proceedings of the Sixteenth Annual*

Allerton Conference on Communication, Control, and Computing, pages 161–173, 1978.

[92] Haruka Mizuta, Takehiro Ito, and Xiao Zhou. Reconfiguration of steiner trees in an unweighted graph. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 100(7):1532–1540, 2017.

[93] G. Morton and A. H. Land. A contribution to the traveling-salesman problem. *Journal of the Royal Statistical Society, Series B*, 17:185–194, 1942.

[94] Amer E. Mouawad. *On Reconfiguration Problems: Structure and Tractability.* PhD thesis, University of Waterloo, 2015.

[95] Amer E. Mouawad, Naomi Nishimura, Venkatesh Raman, Narges Simjour, and Akira Suzuki. On the parameterized complexity of reconfiguration problems. *Algorithmica*, 78(1):274–297, 2017.

[96] Pierre Muller, Jean-Yves Hascoet, and Pascal Mognol. Toolpaths for additive manufacturing of functionally graded materials (fgm) parts. *Rapid Prototyping Journal*, 20(6):511–522, 2014.

[97] Basil R. Myers. Enumeration of tours in hamiltonian rectangular lattice graphs. *Mathematics Magazine*, 54(1):19–23, 1981.

[98] C. M. Mynhardt, L. E. Teshima, and A. Roux. Connected k-dominating graphs. *Discrete Mathematics*, 342(1):145–151, 2019.

[99] Heinrich Niederhausen. How many paths cross at least i given lattice point. *Congressus Numerantium*, 36:161–173, 1982.

[100] Rahnuma Islam Nishat and Sue Whitesides. Bend complexity and hamiltonian cycles in grid graphs. In Yixin Cao and Jianer Chen, editors, *Proceedings of the 23rd International Computing and Combinatorics Conference (COCOON), Hong Kong, China*, volume 10392 of *Lecture Notes in Computer Science*, pages 445–456. Springer, 2017.

[101] Rahnuma Islam Nishat and Sue Whitesides. Reconfiguring hamiltonian cycles in l-shaped grid graphs. In Ignasi Sau and Dimitrios M. Thilikos, editors, *Proceedings of the 45th International Workshop on Graph-Theoretic Concepts*

in Computer Science (WG), Vall de Núria, Spain, volume 11789 of *Lecture Notes in Computer Science*, pages 325–337. Springer, 2019.

[102] Naomi Nishimura. Introduction to reconfiguration. *Algorithms*, 11(4):52, 2018.

[103] Oystein Ore. A note on hamiltonian circuits. *American Mathematical Monthly*, 67:55, 1960.

[104] Hiroki Osawa, Akira Suzuki, Takehiro Ito, and Xiao Zhou. The complexity of (list) edge-coloring reconfiguration problem. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 101(1):232–238, 2018.

[105] Christos H. Papadimitriou. The euclidean travelling salesman problem is np-complete. *Theoretical Computer Science*, 4(3):237 – 244, 1977.

[106] Ian Parberry. An efficient algorithm for the knight's tour problem. *Discrete Applied Mathematics*, 73(3):251 – 260, 1997.

[107] Ville Pettersson. Enumerating hamiltonian cycles. *The Electronic Journal of Combinatorics*, 21(4):P4.7, 2014.

[108] J. Plesńik. The np-completeness of the hamiltonian cycle problem in planar diagraphs with degree bound two. *Information Processing Letters*, 8(4):199–201, 1979.

[109] L. Pósa. A theorem concerning hamiltonian lines. *Magyar Tud. Akad. Mat. Kutatd Int. K6zl.*, 7:225–226, 1962.

[110] M. Sohel Rahman and Mohammad Kaykobad. On hamiltonian cycles and hamiltonian paths. *Information Processing Letters*, 94(1):37–41, 2005.

[111] John R. Reay and Tudor Zamfirescu. Hamiltonian cycles in t-graphs. *Discrete & Computational Geometry*, 24(2-3):497–502, 2000.

[112] Frank Ruskey and Joe Sawada. Bent hamilton cycles in d-dimensional grid graphs. *The Electronic Journal of Combinatorics*, 10, 2003.

[113] Alexander Schrijver. On the history of combinatorial optimization (till 1960). In K. Aardal, G. L. Nemhauser, and R. Weismantel, editors, *Discrete Optimization*, volume 12 of *Handbooks in Operations Research and Management Science*, pages 1 – 68. Elsevier, 2005.

[114] Scott Sheffield. Computing and sampling restricted vertex degree subgraphs and hamiltonian cycles. *CoRR*, 2000.

[115] Robert Elliott Stoyan and Volker Strehl. Enumeration of hamiltonian circuits in rectangular grids. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 1996.

[116] Asahi Takaoka. Complexity of hamiltonian cycle reconfiguration. *Algorithms*, 11(9):140, 2018.

[117] Yoshiyasu Takefuji. *Knight's Tour Problems*, pages 111–118. Springer US, Boston, MA, 1992.

[118] W. T. Tutte. On Hamiltonian Circuits. *Journal of the London Mathematical Society*, s1-21(2):98–101, 1946.

[119] Christopher Umans and William Lenhart. Hamiltonian cycles in solid grid graphs. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS), Miami Beach, Florida, USA*, pages 496–505. IEEE Computer Society, 1997.

[120] Christopher M. Umans. An Algorithm for Finding Hamiltonian Cycles in Grid Graphs Without Holes. Bachelor of Arts with honors thesis, Williams College, Williamstown, Massachusetts, USA, 1996.

[121] Jan van den Heuvel. The complexity of change. *Surveys in Combinatorics*, 409 (2013):127–160, 2013.

[122] K. Wagner. Bemerkungen zum vierfarbenproblem. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 46:26–32, 1936.

[123] Hassler Whitney. Congruent graphs and the connectivity of graphs. In James Eells and Domingo Toledo, editors, *Hassler Whitney Collected Papers*, pages 61–79. Birkhäuser Boston, Boston, MA, 1992.

[124] Stephan Winter. Modeling costs of turns in route planning. *Geoinformatica*, 6 (4):345–361, 2002.

[125] Christina Zamfirescu and Tudor Zamfirescu. Hamiltonian properties of grid graphs. *SIAM Journal on Discrete Mathematics*, 5(4):564–570, 1992.