Reconfiguring Simple s, t Hamiltonian Paths in Rectangular Grid Graphs

Rahnuma Islam Nishat, Venkatesh Srinivasan, and Sue Whitesides

Department of Computer Science, University of Victoria, BC, Canada
{rnishat,srinivas,sue}@uvic.ca

Abstract. We study the following reconfiguration problem: given two s, t Hamiltonian paths connecting diagonally opposite corners s and t of a rectangular grid graph G, can we transform one to the other using only *local* operations in the grid cells? In this work, we introduce the notion of *simple* s, t Hamiltonian paths, and give an algorithm to reconfigure such paths of G in $O(|G|)$ time using local operations in unit grid cells. We achieve our algorithmic result by proving a combinatorial *structure theorem* for simple s, t Hamiltonian paths in rectangular grid graphs.

1 Introduction

An $m \times n$ *rectangular grid graph* G is a subgraph of the infinite integer grid embedded on m rows and n columns; the outer boundary of G is a rectangle $\mathcal{R}$, and the inner faces of G are 1×1 grid cells, so G has mn vertices. An s, t *Hamiltonian path* P of G is a Hamiltonian path of G with endpoints at the top left and bottom right vertices s and t of $\mathcal{R}$. See Figure 2(a) for an example. The *reconfiguration of* s, t *Hamiltonian paths* in grid graphs is concerned with transforming one s, t Hamiltonian path into another such Hamiltonian path of the same graph using some *operation* that preserves Hamiltonicity in each intermediate step of the transformation. Here, we give a reconfiguration algorithm for a class of s, t Hamiltonian paths that we call 'simple' paths. Moreover, we use an operation (namely, *pairs of switch operations* to be defined in Section 4) that is *local* to the grid graph, not the path. See Figure 1.

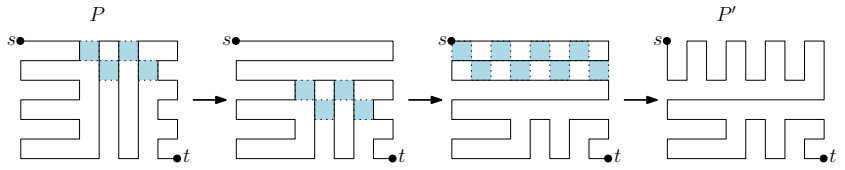

Fig. 1. Reconfiguring simple s, t Hamiltonian path P to another such path P' using pairs of switch operations.

Each internal node v of G lies on an *internal subpath* of P, namely the subpath joining the first boundary vertices v_s and v_t met when travelling along P from v toward s and toward t, respectively. This internal subpath must make at least two bends (turns) if v_s and v_t lie on the same side of $\mathcal{R}$, at least one bend if they lie on adjacent sides, and does not need to bend if v_s and v_t lie directly opposite each other. An s, t Hamiltonian path P is *simple* if each such internal subpath has the minimum possible number of bends. Thus each internal node of G either lies on a bend-free internal subpath between directly opposite boundary nodes, or lies on a one-bend internal subpath between nodes on adjacent sides, or lies on a two-bend internal subpath between nodes on the same side. We call such paths P simple because they cannot wind and twist inside $\mathcal{R}$: internal subpaths can travel between any pair of sides of $\mathcal{R}$, yet can make only the minimum possible number of bends to do so. The Hamiltonian path in Figure 2(a) is simple. See Figure 11 for an example of a 'twisty' and 'windy' s, t Hamiltonian path.

Although linear time reconfiguration algorithms have been designed for "1-complex" Hamiltonian cycles in rectangular grid graphs [**?**] (i.e., Hamiltonian cycles where each internal node of the grid is connected to the boundary by a straight line segment along the cycle), similar techniques do not carry over to s, t Hamiltonian paths. Whereas all the internal paths in a cycle must start and end on adjacent nodes on the boundary, paths allow internal subpaths to have endpoints that are far apart on the boundary. In this paper we study the structure and reconfiguration of simple s, t Hamiltonian paths as an important step towards understanding the structure of general s, t Hamiltonian paths and designing reconfiguration algorithms for them.

Hamiltonian paths and cycles in grid graphs have attracted interest in part due to their many applications (e.g., in robot navigation [**?**], 3D printing [**?**], and polymer science [**?**]). They have the potential to reduce turn costs and travel time and to increase navigation accuracy (e.g., [**?**], [**?**], [**?**]). We believe the general study of reconfiguration of paths in grids is both interesting on its own and also useful for exploring the space of route possibilities in grid-like environments such as warehouses and 3D printer platforms.

Our Contributions. Our work is the first to study reconfiguration of paths in grid graphs, while previous work on paths focused mainly on existence, enumeration, and generation, not reconfiguration. In particular, (1) we establish the structure of simple s, t Hamiltonian paths in rectangular grid graphs; (2) we introduce a *zip* operation, which uses a *switch* in a grid cell as the atomic local operation, to reconfigure simple s, t Hamiltonian paths; our zip operation is comprised of pairs of switch operations that preserve Hamiltonicity and its accompanying data structure facilitate running time analysis and possible implementation of our algorithm and other reconfiguration algorithms; (3) using the structure theorem, we give an algorithm to reconfigure any simple s, t Hamiltonian path to any other such path in time linear in the size $|G|$ of the grid graph G.

Related Work. Itai *et al.* [**?**] gave necessary and sufficient conditions for the existence of a Hamiltonian path between any pair of vertices in a rectangular

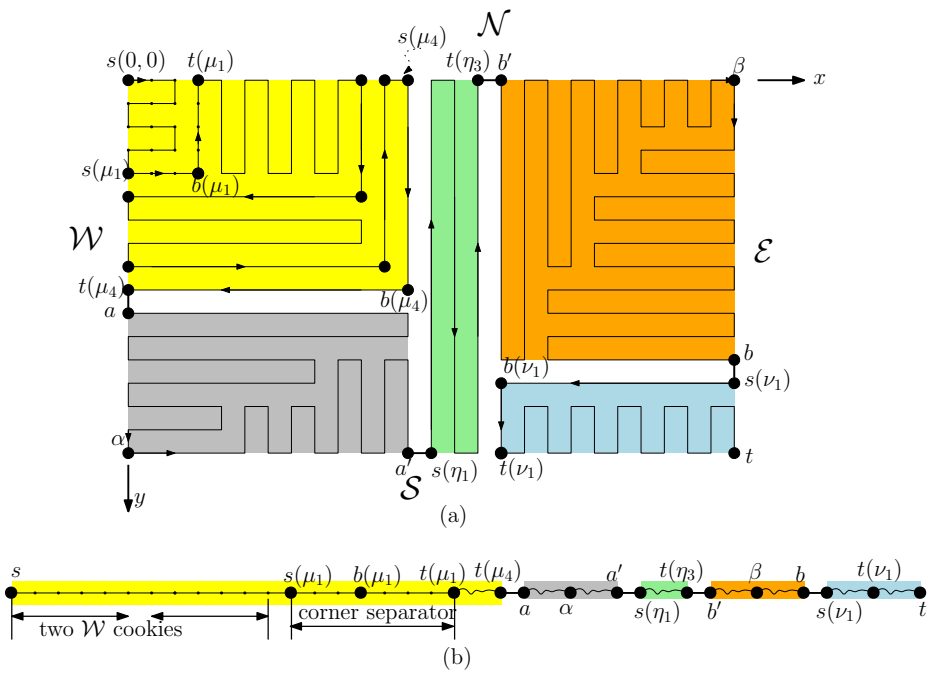

Fig. 2. (a) Grid layout of a simple path $P = P_{s,t}$. (b) The combinatorial layout of P. (See Sections 2 and 3 for explanation of notation.)

grid graph. The existence problem for classes of non-rectangular grid graphs was studied in [?,?,?]. Ruskey and Sawada studied existence of 'bent' Hamiltonian cycles in grid graphs in d-dimension, $d \geq 2$, where each edge in a pair of successive edges on the path lies in a different dimension [?]. Combinatorial aspects of Hamiltonian paths in grid graphs such as enumeration [?,?] and generating functions [?] have been explored.

Reconfiguration problems have attracted attention for some time [?,?]. There has been some recent work on reconfiguration of Hamiltonian cycles in grid graphs. Takaoka [?] has shown that for some unembedded graph classes, deciding whether there is a sequence of "switch" operations between two given Hamiltonian cycles is a PSPACE-complete problem. Nishat and Whitesides studied reconfiguration of Hamiltonian cycles of "bend complexity 1" in grid graphs without holes [?,?,?].

2 Terminology and Basics

Throughout this paper, a *simple path* means a simple s, t Hamiltonian path of G; P visits each node of G exactly once and uses only edges in G. A *cell* of G is an internal face of G. A vertex of G with coordinates (x, y) is denoted by $v_{x,y}$,

where $0 \leq x \leq n-1$ and $0 \leq y \leq m-1$. The top left corner vertex s of G has coordinates $(0,0)$, and the positive y-direction is downward. We use the two terms *node* and *vertex* interchangeably.

Column x of G is the shortest path of G between $v_{x,0}$ and $v_{x,m-1}$, and *Row* y is the shortest path between $v_{0,y}$ and $v_{n-1,y}$. We call Columns 0 and $n-1$ the *west* ($\mathcal{W}$) and *east* ($\mathcal{E}$) boundaries of G, respectively, and Rows 0 and $m-1$ the *north* ($\mathcal{N}$) and *south* ($\mathcal{S}$) boundaries.

Let P be a simple path of G. We denote by $P_{u,w}$ the directed subpath of P from vertex u to w. Straight subpaths are called segments, denoted $seg[u,v]$, where u and v are the segment endpoints. An internal subpath $P_{u,v}$ of P is called a *cookie* if both u, v are on the same boundary (i.e., $\mathcal{N}$, $\mathcal{S}$, $\mathcal{E}$, and $\mathcal{W}$); otherwise, $P_{u,v}$ is called a *separator*. (Note that removal of the nodes of a separator from G separates s from t in G.)

Cookies and Separators. A cookie has one of four types, depending on the boundary where the cookie has its *base*. A cookie c is formed by three segments of P; the common length of the two parallel segments measures the *size* of c.

Since separators of P have endpoints on distinct boundaries, there are two kinds, as shown in Figure 2: a *corner separator* μ_i or ν_i has one bend, and a *straight separator* η_i has no bends. Traveling along $P_{s,t}$, we denote the i-th straight separator we meet by η_i where $0 \leq i \leq k$, and its endpoints by $s(\eta_i)$ and $t(\eta_i)$, where $s(\eta_i)$ is the first endpoint met. We say a corner separator *cuts off* a corner (s or t). Traveling along $P_{s,t}$, we denote the i-th corner separator cutting off s by μ_i, where $0 \leq i \leq j$. We denote its internal bend by $b(\mu_i)$, and its endpoints by $s(\mu_i)$ and $t(\mu_i)$, where $s(\mu_i)$ is the first endpoint met. Similarly, we denote the i-th corner separator cutting off t by ν_i; endpoint $s(\nu_i)$ is met before $t(\nu_i)$, with internal bend at $b(\nu_i)$, where $0 \leq i \leq \ell$. A corner separator that has one of its endpoint connected to s or t by a segment of P is called a *corner cookie*. We have j corner separators μ_i cutting off s, and k straight separators, and ℓ corner separators ν_i cutting off t. In Figure 2, j=4, k= 3 and ℓ=1. (We will see that only s and t can be cut off.)

Runs of Cookies. A *run of cookies* is a subpath of P consisting of cookies of the same type, spaced one unit apart and joined by the single boundary edges between them, possibly extended at either end by an edge joining a cookie endpoint to an adjacent boundary vertex. A run of cookies is denoted $Run[u,v]$, where u and v belong to the same boundary and delimit the range of boundary vertices covered; $Run[u,v]$ may consist of a single boundary edge (u,v). In Figure 2, $Run[s, s(\mu_1)]$ is a run of $\mathcal{W}$ cookies. To describe the path structure, we define three types of runs, depending on the cookie sizes along the run: the sizes may remain the same, or be non-increasing (denoted $Run^{\geq}[u,v]$) or non-decreasing ($Run^{\leq}[u,v]$). Runs are assumed to have cookies of the same size unless specified otherwise. In Figure 2, $Run^{\geq}[a,\alpha]$ is non-increasing; $Run[s, s(\mu_1)]$, which has same size cookies, could also be viewed as non-increasing or non-decreasing.

Canonical Paths. A *canonical path* is a simple path P with no bends at internal vertices. If m is odd, P can be $\mathcal{E}$-$\mathcal{W}$ and fill rows of G one by one; if n is odd, P can be $\mathcal{N}$-$\mathcal{S}$ and fill columns (see Figure 3(d)). There are no other types.

Assumption. Let α and β denote the bottom left and top right corner vertices of G. Without loss of generality, *we assume the input simple path $P_{s,t}$ visits α before β*. The target simple path for the reconfiguration as well as intermediate configurations may visit β before α. In the rest of this section and the next, P denotes the input simple path.

Blocks. As shown in Figure 3, corner separators μ_j and ν_1 define rectangular subgraphs, called *blocks*, of G, denoted $\mathcal{R}_s$ and $\mathcal{R}_t$. Straight separators η_1 and η_k determine a block $\mathcal{R}_{mid}$. The remaining nodes of G determine two blocks: $\mathcal{R}_\alpha$, with corners at α, a (upper left) and a' (lower right), and $\mathcal{R}_\beta$, with corners at β, and b' (upper left) and b (lower right) as shown. Blocks $\mathcal{R}_s$ and $\mathcal{R}_t$ vanish when $j{=}0$ and $\ell{=}0$, respectively. For now the goal is to define the blocks. Later we will observe that $P_{s,t}$ connects the blocks with "links" as shown in the figure.

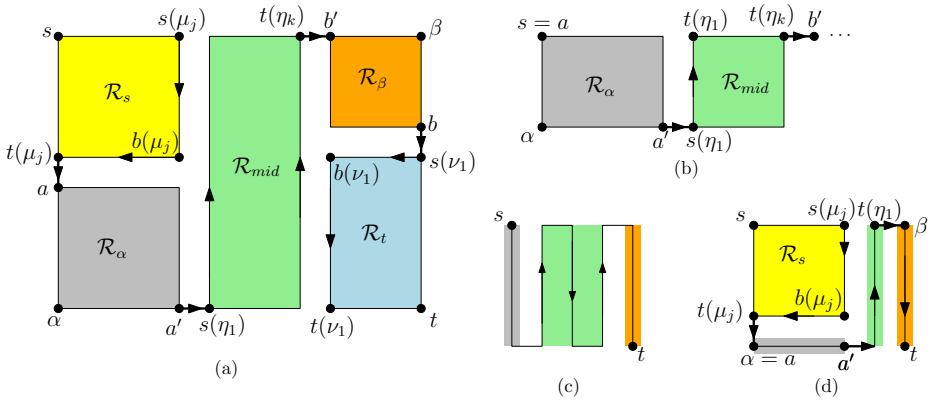

Fig. 3. (a) Blocks of $P_{s,t}$; the *links* between blocks are darkened. (b)–(d) Some special cases. See Section 3 for details. (c) An $\mathcal{N}$-$\mathcal{S}$ canonical path.

We make two observations (the second is a consequence of the first) that prepare for the structure theorem in the next section.

Observation 1. The subpath $P_{s,\alpha}$ of $P_{s,t}$ must cover all the $\mathcal{W}$ vertices as otherwise P fails to be Hamiltonian or non-crossing. For the same reason, no corner separators cut off α or β, and $P_{s,t}$ must visit β before visiting any other $\mathcal{E}$ vertices. It follows that the corner separators μ_i cutting off s must occur in $P_{s,t}$ before α, and that the corner separators ν_i cutting off t occur in $P_{s,t}$ after β. It also follows that all straight separators η_i occur between α and β, and that the number $k \geq 1$ of them must be odd. □

Observation 2. $P_{s,t}$ breaks G into at most five blocks ($\mathcal{R}_s$, $\mathcal{R}_\alpha$, $\mathcal{R}_{mid}$, $\mathcal{R}_\beta$, and $\mathcal{R}_t$) as shown in Figure 3. $P_{s,t}$ must join blocks with *link* edges, one on each boundary, directed as shown in the figure. In certain cases, detailed in Section 3, blocks (and their links) may vanish, or blocks may shrink to segments. □

3 Structure of a Simple Path P

We regard P in its directed form $P_{s,t}$ as composed of an *initial* subpath $P_{s,s(\eta_1)}$, followed by a *middle* subpath $P_{s(\eta_1),t(\eta_k)}$, and a *final* subpath $P_{t(\eta_k),t}$. Except for three special cases **(a)-(c)** listed below, the initial subpath first covers all the vertices in $\mathcal{R}_s$, next takes an edge $(t(\mu_j), a)$, denoted $link(t(\mu_j), a)$, to $\mathcal{R}_\alpha$. It then covers $\mathcal{R}_\alpha$ and then takes an edge $link(a', s(\eta_1))$ to $s(\eta_1)$. The possible forms for the final subpath are the same as for the initial subpath $P_{t,s}$, which is $P_{s,t}$ in reverse.

Three special cases: (a) If α is not adjacent to $s(\eta_1)$ in G and $j = 0$, then $s = a$ because $\mathcal{R}_s$ disappears and the initial subpath begins at $\mathcal{R}_\alpha$, where each side of $\mathcal{R}_\alpha$ has length at least 1. See Figure 3(b). Similarly, if β is not adjacent to $t(\eta_k)$ and $\ell = 0$, then $\mathcal{R}_t$ disappears and $b = t$ and each side of $\mathcal{R}_\beta$ has length at least 1. **(b)** If α is adjacent to $s(\eta_1)$ in G and thus in $P_{s,t}$, then there is no room for any μ_i so as in case (a), j must be 0 and $a = s$; furthermore, unlike case (a), $\mathcal{R}_\alpha = seg[s = a, \alpha]$. See Figure 3(c) for an example. Similary, if $t(\eta_k)$ is adjacent to β, then ℓ must be 0 and $b = t$, and $\mathcal{R}_\beta = seg[\beta, b = t]$. **(c)** If $j > 0$ and $P_{s,t}$ contains edge $(t(\mu_j), \alpha)$ then $\mathcal{R}_\alpha = seg[\alpha, a']$. See Figure 3(d). Similarly, if $\ell > 0$ and $P_{s,t}$ contains edge $(\beta, s(\nu_1))$, then $\mathcal{R}_\beta = seg[b', \beta]$.

A canonical path that visits α before β falls into case **(b)** and has the form $\| \; seg[s = a, \alpha] \; \| \; link(\alpha, s(\eta_1)) \; \| \; P_{s(\eta_1),t(\eta_k)} \; \| \; link(t(\eta_k), \beta) \; \| \; seg[\beta, b = t] \; \|$. (Recall Figure 3(c).) Here, and later on, $\|$ is used to delimit subpaths. Observations 3, 4, and 5 below, together with Observations 1, 2 and associated figures, will establish our structure theorem.

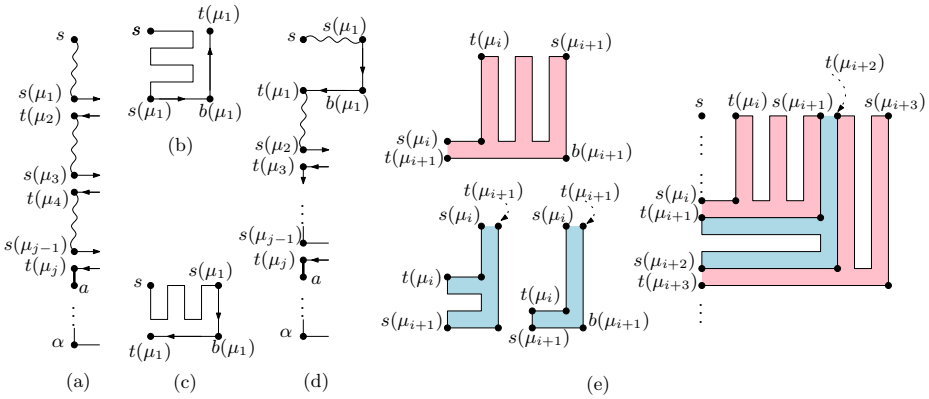

Fig. 4. Endpoint ordering on $\mathcal{W}$ and form of $P_{s,s(\mu_1)}$ for: (a),(b) even $j > 0$; (c),(d) j odd; (e) round trips $P_{s(\mu_i),t(\mu_{i+1})}$ start on $\mathcal{W}$ (red) or on $\mathcal{N}$ (blue).

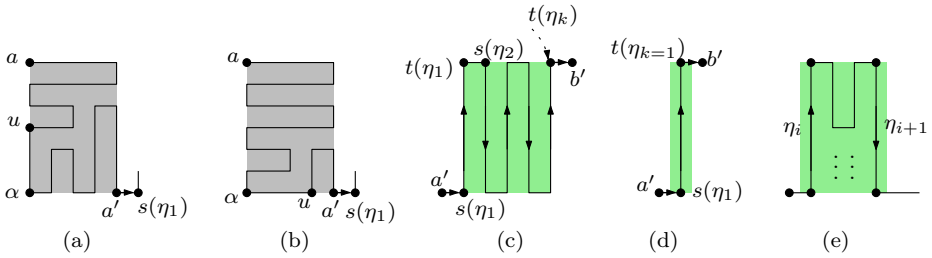

Fig. 5. (a) Block $\mathcal{R}_\alpha$ where $|seg[u,\alpha]| > 1$ and (b) where $|seg[\alpha,u]| > 1$; (c) $\mathcal{R}_{mid}$ for $k > 1$; (d) $\mathcal{R}_{mid}$ is a segment when $k = 1$. (e) $\mathcal{R}_{mid}$ cannot have cookies.

Observation 3. (about $\mathcal{R}_s$) For $j = 0$, $\mathcal{R}_s$ disappears. For $j > 0$, there must be j endpoints of μ_i on $\mathcal{W}$ and j on $\mathcal{N}$. Endpoint $t(\mu_j)$ must lie on $\mathcal{W}$, not $\mathcal{N}$, and it must be the corner separator endpoint nearest to α on $\mathcal{W}$. The endpoints must alternate on $\mathcal{W}$ as shown in Figures 4(a),(d). Thus for j odd, the top endpoint on $\mathcal{W}$ is $t(\mu_1)$, so $s(\mu_1)$ is on $\mathcal{N}$; for $j > 0$ even, $s(\mu_1)$ is on $\mathcal{W}$. Either way, $P_{s,s(\mu_1)} = Run[s, s(\mu_1)]$.

$P_{s(\mu_1),t(\mu_j)}$ makes round trips (see Figure 4(e)) between $\mathcal{W}$ and $\mathcal{N}$ via the μ_i. These trips have the form $\|\mu_i \| Run[t(\mu_i), s(\mu_{i+1})] \|\mu_{i+1}\|$ and alternate leaving from $\mathcal{N}$ or from $\mathcal{W}$, as the return leg of one trip is the outgoing leg of the next. Furthermore, $t(\mu_{i+1})$ must be adjacent to $s(\mu_i)$ to ensure the Hamiltonicity of $P_{s,t}$. Between $t(\mu_i)$ and $s(\mu_{i+1})$, the path must have the form $Run[t(\mu_i), s(\mu_{i+1})]$, which may be just a single edge $(t(\mu_i), s(\mu_{i+1}))$.

Thus for $j > 0$, $P_{s,t(\mu_j)} = \|Run[s, s(\mu_1)] \| \mu_1 \| Run[t(\mu_1), s(\mu_2)] \| \mu_2 \| \ldots \| \mu_i \| Run[t(\mu_i), s(\mu_{i+1})] \| \mu_{i+1} \| \ldots \| \mu_{j-1} \| Run[t(\mu_{j-1}), s(\mu_j)] \| \mu_j\|$. □

Observation 4. (about $\mathcal{R}_\alpha$) In special case **(b)**, in which $s = a$ and α is adjacent to $s(\eta_1)$, we have that $\alpha = a'$ and so $P_{a,a'} = seg[s = a, \alpha]$ on $\mathcal{W}$. In special case **(c)**, in which $P_{s,t}$ contains edge $(t(\mu_j), a = \alpha)$, we have that $P_{a,a'} = seg[a = \alpha, a']$ on $\mathcal{S}$. Otherwise, two segments of P meet at α, one on $\mathcal{W}$, one on $\mathcal{S}$; one has unit length and the other is strictly longer. See Figure 5(a),(b). It follows from Observation 1 that either $P_{a,a'} = \|Run^{\geq}[a, u] \| seg[u, \alpha] \| Run^{\leq}[\alpha, a']\|$, with u at least two units from α on $\mathcal{W}$, or $P_{a,a'} = \|Run^{\geq}[a, \alpha] \| seg[\alpha, u] \| Run^{\leq}[u, a']\|$, with u at least two units from α on $\mathcal{S}$. □

Observation 5. (about $\mathcal{R}_{mid}$) (See Figure 5(c),(d),(e).) Block $\mathcal{R}_{mid}$ has no cookies. Extending run notation, $P_{s(\eta_1),t(\eta_k)} = \|\eta_1 \| Run[t(\eta_1), t(\eta_k)]\|$. □

The next theorem, based on Observations 1–5 and associated figures, establishes the structure of P by giving the forms that the subpaths of $P_{s,t}$ may take inside the blocks. See Figure 3.

Theorem 1 (Structure of Simple Paths). *Let $P_{s,t}$ be a simple path with k straight separators and j and ℓ corner separators cutting off s and t, respectively.*

initial subpath $[P_{s,s(\eta_1)}]$ *The subpath* $P_{s,t(\mu_j)}$ *through* $\mathcal{R}_s$ *is given in Observation 3. The subsubpath* $P_{a,a'}$ *through* $\mathcal{R}_\alpha$ *is given in Observation 4. Appending* $link(a', s(\eta_1))$ *and inserting* $link(t(\mu_j), a)$ *(if needed), gives the structure for* $P_{s,s(\eta_1)}$.

middle subpath $[P_{s(\eta_1),t(\eta_k)}]$ *This consists of an odd number* k *of straight separators* η_i *in adjacent grid columns, joined by edges* $(t(\eta_i), s(\eta_{i+1}))$, $1 \leq i < k$, *which lie on* N *for odd* i *and on* S *for even* i. *See Observation 5.*

final subpath $[P_{t(\eta_k),t}]$ *As* $P_{t,t(\eta_k)}$ *is the initial subpath of* $P_{t,s}$ *(the reverse of* $P_{s,t}$*), the forms for the final and the initial subpaths are the same.*

4 Zip operation

In this section we define the *zip* operation that can be applied on an s, t Hamiltonian path P, not necessarily simple. We will use zips in our reconfiguration algorithm in the next section. We first define some terminology, including a *switch* operation that is used as the atomic local operation of *zip*.

A *vertical track* tr^v_x is the subgraph of G induced by Columns x and $x+1$, and a *horizontal track* tr^h_y is the subgraph induced by Rows y and $y+1$. A *cycle-path cover* $\mathbb{P}$ of G is a set of cycles and paths that collectively cover all the vertices of G. Let f be a cell of G with face cycle a, b, c, d, a in G such that edges (a, d) and (c, b) are the only edges of f that are in $\mathbb{P}$ (see Figure 6). A *switch* operation on $\mathbb{P}$ in f replaces (a, d) and (c, b) with (a, b) and (c, d) and thus produces another cycle-path cover $\mathbb{P}'$. We call f a *switchable* cell of G in $\mathbb{P}$ (or in P for short, if $\mathbb{P} = \{P\}$). Note that our switch operation is applied locally to a cell of an *embedded* grid graph, and hence a single switch does not preserve Hamiltonicity. We, therefore, always apply switches in pairs such that the second switch operation in a pair patches the two parts created by the first switch and returns an s, t Hamiltonian path. In contrast, the switch operation defined by Lignos [**?**] and Takaoka [**?**] is applied to a four cycle of the graph, where the graph is not necessarily planar, and may not be embedded; and their switch operations do not need to be paired. See Figure 6(c) for an example of the switch operation defined in [**?**,**?**].

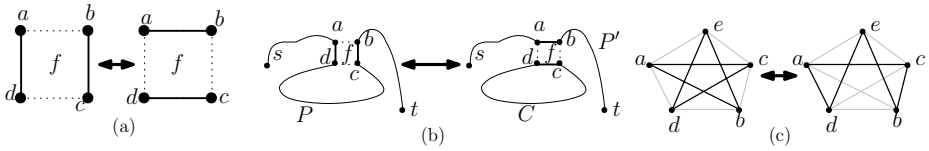

Fig. 6. (a) A *switch* in a cell f. (b) A switch in a switchable cell of an *embedded* s, t Hamiltonian path P in cycle-path cover $\mathbb{P} = \{P\}$ yields new cover $\mathbb{P}' = \{P', C\}$. (c) A single switch (i.e., edges (a, b) and (d, c) is replaced by edges (a, d) and (b, c)) as defined in [**?**,**?**] on a non-planar graph preserves Hamiltonicity.

Observation 6. A switch operation in a grid cell that is switchable for an s, t Hamiltonian path P of G gives a cycle-path cover of G: $\mathbb{P}' = \{C, P'\}$ where C is a cycle and P' is a path with ends s and t. A switch in a switchable cell f of $\mathbb{P}'$, where each of C and P' contains one edge of f, gives an s, t Hamiltonian path.

A *zipline* $l_z^{q_1,q_2}$ (the superscript may be omitted for short) is a straight directed line in G from node q_1 to node q_2; l_z lies in a row or a column of G, and two tracks (horizontal or vertical) of G contain l_z. We designate one of those tracks as the *cookie track* tr, and the other track the *side track* tr'. A switchable cell f of tr or tr' is called a *perpendicular switchable cell* if the two edges that f contributes to P are perpendicular to tr (and tr'); we denote the first of those edges met while walking along l_z from q_1 to q_2 by $e_1(f)$, the second edge by $e_2(f)$, and we say that f is *between* q_1 *and* q_2 if $e_1(f)$ and $e_2(f)$ occur between q_1 and q_2. Note that not all the switchable cells of tr and tr' between q_1 and q_2 are necessarily perpendicular switchable cells.

The *zip set* S of zipline l_z is the set of perpendicular switchable cells from tr and tr' constructed as follows: walking along l_z from q_1 to q_2, we look for the first perpendicular switchable cell in the cookie track tr; if we find such a switchable cell, say f_1, in tr between q_1 and q_2, we look for the next perpendicular switchable cell in the side track tr' between $e_2(f_1)$ and vertex q_2; if we find such a cell (call it f_1') in tr' then we add both f_1 and f_1' to S; otherwise we do not add f_1 to S as it cannot be paired with a switchable cell from the side track. If we find f_1' before reaching q_2, we repeat the above process for cells between $e_2(f_1')$ and q_2 to find pairs (f_2, f_2'), (f_3, f_3') and so on. At the end, S is either empty or consists of pairs of perpendicular switchable cells, one from tr and one from tr'. Moreover, no two cells of S share an edge, and each pair f_i and f_i' is met consecutively on l_z between f_{i-1}' and f_{i+1}. We now define the *zip* operation; see Figure 7.

Definition 1 (Zip). *Let $l_z^{q_1,q_2}$ be a zipline of G with cookie track tr and side track tr', and let $S = \{f_1, f_1', f_2, f_2', \ldots, f_p, f_p'\}$, where $|S| \geq 0$ is even, be the zip set of l_z. The* zip *operation $Z = zip(l_z^{q_1,q_2}, tr, tr')$ applies switch to all the cells of S in the following order: $f_1', f_1, f_2', f_2, \ldots, f_p', f_p$.*

We note here that the zip operation may not preserve simplicity when applied on a simple path. However, as Lemma 1 shows, each zip and indeed each pair of its switches, preserves s, t Hamiltonicity.

Let $S = \{f_1, f_1', f_2, f_2', \ldots, f_p, f_p'\}$ be the zip set of the zip operation $Z = zip(l_z^{q_1,q_2}, tr, tr')$, where $|S| > 0$. S is *intra-pair overlapping* if for each pair of cells f_i, f_i' of S, the edges appear in the following *path order* on $P_{s,t}$: $e_1(f_1)$, $e_1(f_1')$, $e_2(f_1)$, $e_2(f_1')$. Note that the path order of those four edges is different from their *zipline order*, i.e., the order in which they are met on l_z. Zip set S is *inter-pair overlapping* when S is intra-pair overlapping, and the first edge $e_1(f_{i+1})$ of the $(i+1)^{st}$ pair is between $e_2(f_i)$ and $e_2(f_i')$ on P. In other words, the path order of the edges of the cells of S is $e_1(f_1)$, $e_1(f_1')$, $e_2(f_1)$, $e_1(f_2)$, $e_2(f_1')$, $e_1(f_2')$, $e_2(f_2)$, $e_1(f_3)$, $e_2(f_2')$, $\ldots$, $e_2(f_p')$.

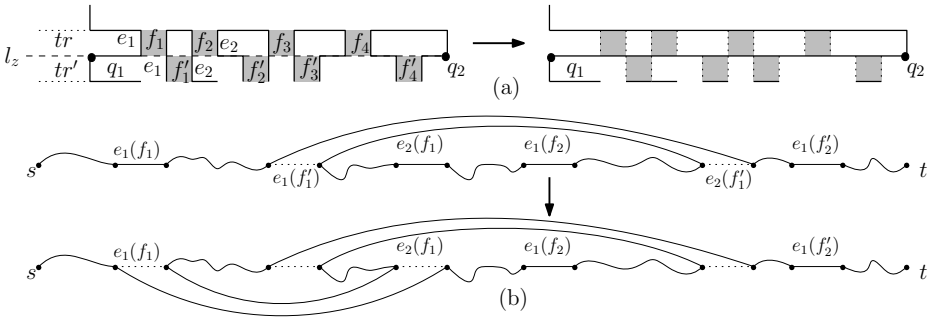

Fig. 7. (a) An example of a zip operation on an s, t Hamiltonian path P where l_z is horizontal. (b) The cycle-path cover $\mathbb{P}$ obtained by a switch in f'_1, and the new s, t Hamiltonian path P' obtained from the cycle-path cover by a switch in f_1.

Lemma 1. *Let $Z = zip(l_z^{q_1,q_2}, tr, tr')$ be a zip operation on an s, t Hamiltonian path P with the zip set $S = \{f_i, f'_i$ for $1 \leq i \leq p\}$, $|S| > 0$. If S is inter-pair overlapping, then after switching the cells f'_1 and f_1 we obtain an s, t Hamiltonian path P' such that $S' = S - \{f_1, f'_1\}$ is the zip set of $Z' = zip(l_z^{q,q_2}, tr, tr')$, where q is the endpoint of $e_2(f'_1)$ on l_z, and S' is inter-pair overlapping.*

Proof. We first prove that P' is an s, t Hamiltonian path. Since S is intra-pair overlapping, the path order of the edges of f_1 and f'_1 is $e_1(f_1)$, $e_1(f'_1)$, $e_2(f_1)$, $e_2(f'_1)$. By Observation 6, a switch in f'_1 must produce a cycle-path cover, say $\mathbb{P}$, of G containing exactly one cycle and one path from s to t; see Figure 7(b). Now edge $e_1(f_1)$ is on the path and the edge $e_2(f_1)$ is on the cycle in $\mathbb{P}$. Therefore, the switch in f_1 in $\mathbb{P}$ gives an s, t Hamiltonian path by Observation 6. Since by definition of zip set, f_2 is the first perpendicular switchable cell encountered in cookie track tr when walking from $e_2(f'_1)$ to q_2, $S' = S - \{f_1, f'_1\}$ is the zip set of Z'. By the inter-pair overlapping property of S, only one edge $e_1(f_2)$ of a cell in $S - \{f_1, f'_1\}$ is on the subpath of P from s to the second endpoint of $e_2(f'_1)$. Since P and P' only differ in that subpath, the path order of the edges of the cells of S' on P' is the same as their path order on P. Therefore, S' is inter-pair overlapping. □

Using Lemma 1, we can prove the following lemma by induction. To achieve the stated time complexity, a suitable data structure is used to store the path edges that allows $O(1)$ time retrieval and update.

Lemma 2. *A zip operation on an s, t Hamiltonian path P of G returns another s, t Hamiltonian path P' in $O(\max\{m, n\})$ time, where P and P' differ only at the cells belonging to the zip set.*

5 Reconfiguring Simple Paths

In this section, we give an algorithm to reconfigure any simple path P to another simple path P'. The algorithm reconfigures P and P' to canonical paths $\mathbb{P}$ and $\mathbb{P}'$, respectively; it then reconfigures $\mathbb{P}$ to $\mathbb{P}'$ in case they are not the same path (i.e., one of $\mathbb{P}$ and $\mathbb{P}'$ is the $\mathcal{N}$-$\mathcal{S}$ canonical path and the other is the $\mathcal{E}$-$\mathcal{W}$ canonical path); and finally the algorithm reverses the steps taken from P' to $\mathbb{P}'$.

Reconfiguring P to $\mathbb{P}$: We give an algorithm that we call ReconfigSimp to reconfigure any simple path $P = P_{s,t}$ to a canonical path $\mathbb{P}$, where $\mathbb{P}$ might be either $\mathcal{N}$-$\mathcal{S}$ or $\mathcal{E}$-$\mathcal{W}$. The algorithm runs in three steps: (a) reconfigure the initial subpath of $P_{s,t}$ such that either $P_{s,\alpha}$ is a segment $seg[s,\alpha]$ and $\alpha = a'$, or $Run[s,\alpha]$ is a run of $\mathcal{W}$ cookies of size $x(a')$ and $P_{\alpha,a'}$ is a segment $seg[\alpha,a']$; (b) reconfigure the final subpath (that is, the initial subpath of $P_{t,s}$) similarly to the previous step; (c) if the path resulting from Step (b) contains both the segments $seg[s,\alpha]$ and $seg[\beta,t]$ then we have the $\mathcal{N}$-$\mathcal{S}$ canonical path and the algorithm terminates; otherwise, we have at least one run of $\mathcal{E}$ or $\mathcal{W}$ cookies and we reconfigure $P_{s,t}$ to obtain the $\mathcal{E}$-$\mathcal{W}$ canonical path. (

Step (a). The algorithm reconfigures the initial subpath of P by calling two procedures, first Recon_R_s and then Recon_s_to_a'.

Procedure Recon_R_s reconfigures R_s by dissolving an even number of corner separators cutting off s. If the number j is even, then all the corner separators are dissolved; otherwise, all but μ_j are dissolved.

If $j > 0$ is even, we apply zips on a horizontal zipline $l_z^{q_1,q_2}$, where q_1 and q_2 are on the $\mathcal{W}$ boundary and on Column $x(s(\mu_j))$, respectively, for all the zip operations. The zipline l_z is first placed on Row 1. Track tr_0^h above l_z is the cookie track and track tr_1^h below l_z is the side track. The zip operation then grows a $\mathcal{W}$ cookie of size $x(q_2)$ in the cookie track. We then move the zipline two rows down and apply a similar zip. In this way the zipline is *swept downward* until it is on Row $y(t(\mu_j))$, where the final zip is applied to obtain a simple path P' with no corner separators cutting off s. We call this procedure a SweepDown; see Figure 8.

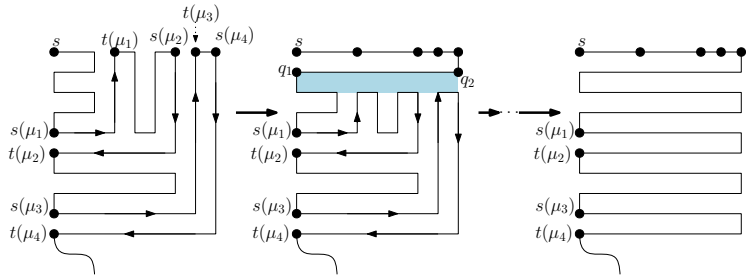

Fig. 8. Intermediate steps of a SweepDown procedure on $\mathcal{R}_s$ of P.

If j is odd, we apply a procedure SweepRight similar to SweepDown. Zips are applied on a vertical zipline $l_z^{q_1,q_2}$, where q_1 and q_2 are on the $\mathcal{N}$ boundary

and on Row $y(t(\mu_j))$, respectively, and the cookie track is on the left and the side track is on the right of l_z. The zipline starts from Column 1 and finishes on Column $x(s(\mu_j)) - 1$, sweeping two columns eastward after every zip except the last one.

Let P' be the path returned by RECON_R_s. The following lemma shows that P' is a simple path with at most one corner separator cutting off s.

Lemma 3. *P' is a simple path with no corner separator if j is even, and P' has a single corner separator that coincides with μ_j of P if j is odd.*

Proof. By Lemma 2, each of the zip operations in the SWEEPDOWN procedure gives an s, t Hamiltonian path, although it might not necessarily be simple. First assume that j is even. To prove that the P' is a simple s, t Hamiltonian path, we show that each zip operation satisfies the following pre- and post-conditions.
Pre-condition: P_{q_1,q_2} occupies only the rows on or below the cookie track.
Post-condition: Cookie track tr contains a $\mathcal{W}$ cookie of size $x(q_2)$.

Since the cookie track of the first zip Z_1 is incident to the $\mathcal{N}$ boundary, the input P satisfies the pre-condition. Let P_1 be the path obtained by applying Z_1 to P. Every switch in tr' creates an island in tr and the next switch in tr connects that island to the $\mathcal{W}$ cookie, thus increasing the size of the $\mathcal{W}$ cookie. The first switch of Z_1 is illustrated in Figure 9. At the end of Z_1, there is one

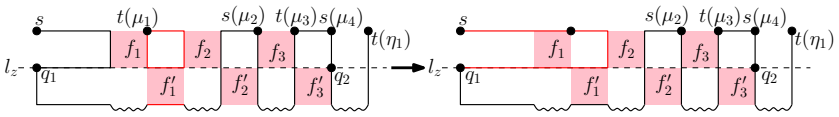

Fig. 9. The 1×1 island, depicted by red line, in the cookie track created by the switch in f'_1 is attached to the $\mathcal{W}$ cookie by the switch in f_1. The W cookie after the switch is depicted by red line.

single $\mathcal{W}$ cookie of size $x(q_2)$ in P_1 satisfying the post-condition. Then P_1 also satisfies the pre-condition of the next zip operation where the cookie track is tr_2^h, as there are no edges perpendicular to the side track tr_1^h of Z_1 in P_1. In this way, we can show that after each zip operation we have a $\mathcal{W}$ cookie of size $x(q_2)$ in the respective cookie track. Therefore, at the end of the sweep, $\mathcal{R}_s$ of P' is covered by a run of $\mathcal{W}$ cookies; see Figure 8. The case when j is odd is the same when the grid is flipped along the s, t diagonal. □

Procedure RECON_s_*to*_a' applies zips on P' to obtain another simple path P'', where the initial subpath of P'' has either a run of $\mathcal{W}$ cookies or no cookies at all. If there is an odd number of columns, including the $\mathcal{W}$ boundary, to the left of η_1, we apply a *SweepLeft* procedure similar to the *SweepRight* procedure above. The zips are applied on a vertical zipline $l_z^{q_1,q_2}$, where q_1 and q_2 are on the $\mathcal{S}$ boundary and $\mathcal{N}$ boundary, respectively, and the cookie track is on the right and

the side track is on the left of l_z. The zipline starts from Column $x(s(\eta_1)) - 2$ and finishes on Column 1, sweeping two columns westward after every zip except the last one. See Figure 10. Otherwise, when there is an even number of columns

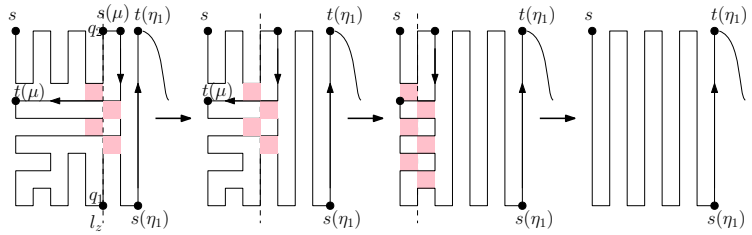

Fig. 10. The intermediate paths in the SweepLeft procedure in P'.

covered by the initial subpath of P, the number of rows must be odd [**?**]. We apply a *SweepDown* procedure from Row 1 to Row $m-2$, where q_1 is on the $\mathcal{W}$ boundary and q_2 is on η_1 for all the ziplines. The following lemma proves the correctness of procedure Recon_s_to_a′.

Lemma 4. *The initial subpath of P'' has either a run of $\mathcal{W}$ cookies or no cookies at all.*

Proof. If $x(\eta_1)$ is even, then there must be an odd number of rows and no corner separators in P'. Then the SweepDown procedure grows a run of $\mathcal{W}$ cookies in a similar way as in the proof of Lemma 3. If $x(\eta_1)$ is odd, then there are two cases to consider based on the existence of a corner separator cutting off s in P'. In both cases, SweepLeft works similarly to SweepDown if we flip the $P_{s,s(\eta_1)}$ subpath about the diagonal. □

Since Step (b) is very similar to Step (a), we now describe Step (c).

Step (c). Let the path obtained after Steps (a) and (b) be $\mathcal{P}$. If there exists any $\mathcal{E}$ or $\mathcal{W}$ cookies in $\mathcal{P}$, then SweepDown is applied on the whole path, where the starting point q_1 of each zipline is on the $\mathcal{W}$ boundary and the ending point q_2 is on the $\mathcal{E}$ boundary. The following theorem follows from Lemmas 3 and 4.

Theorem 2. *Algorithm* ReconfigSimp *reconfigures a simple path in a rectangular grid graph G to a canonical path of G in $O(|G|)$ time.*

Reconfiguring $\mathbb{P}$ to $\mathbb{P}'$: This step is similar to Step (c) of Algorithm ReconfigSimp; if $\mathbb{P}$ is $\mathcal{N}$-$\mathcal{S}$ then we grow horizontal straight separators by sweeping downward; otherwise, we grow vertical separators by sweeping eastward. We now have the following theorem.

Theorem 3. *Let $\mathbb{P}$ and $\mathbb{P}'$ be two different canonical paths of G. Then $\mathbb{P}$ can be reconfigured to $\mathbb{P}'$ using zips in $O(|G|)$ time.*

Main Algorithmic Result: We now state the central algorithmic result of our paper in the following theorem.

Theorem 4. *Let P and P' be two simple paths of a rectangular grid graph G. Then P can be reconfigured to P' using zips in $O(|G|)$ time.*

6 Conclusion

We have opened the exploration of reconfiguration of families of Hamiltonian paths in grid graphs. We have established the structure of any *simple* s, t Hamiltonian path in a rectangular grid graph G and given an $O(|G|)$ algorithm to reconfigure any such path to any other using *zip* operations. It would be interesting to find new families of s, t Hamiltonian paths, as shown in Figure 11, and local operations that can reconfigure them.

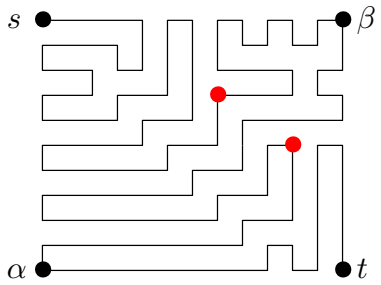

Fig. 11. An s, t Hamiltonian path, where the red vertices are connected to a boundary by at least 4 segments on the path.

The problems about reconfiguring Hamiltonian paths with arbitrary end vertices remain open. Another future direction of research could be reconfiguration of Hamiltonian paths in d dimension, where $d \geq 3$.

References

1. Arkin, E.M., Bender, M.A., Demaine, E.D., Fekete, S.P., Mitchell, J.S.B., Sethia, S.: Optimal covering tours with turn costs. In: Proceedings of the Twelfth Ann. ACM-SIAM Symp. on Discrete Algorithms. pp. 138–147. SODA '01, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2001)
2. Collins, K.L., Krompart, L.B.: The number of Hamiltonian paths in a rectangular grid. Discrete Mathematics **169**(1–3), 29–38 (1997)
3. Everett, H.: Hamiltonian paths in nonrectangular grid graphs. Master's thesis, University of Saskatchewan, Canada (1986)
4. Fellows, M., Giannopoulos, P., Knauer, C., Paul, C., Rosamond, F.A., Whitesides, S., Yu, N.: Milling a graph with turn costs: A parameterized complexity perspective. In: Thilikos, D.M. (ed.) Graph Theoretic Concepts in Computer Science -

372 36th Int. Workshop, WG 2010, Zarós, Crete, Greece, June 28-30, 2010 Revised
373 Papers. Lecture Notes in Computer Science, vol. 6410, pp. 123–134 (2010)
374 5. Gorbenko, A., Popov, V., Sheka, A.: Localization on discrete grid graphs. In: He,
375 X., Hua, E., Lin, Y., Liu, X. (eds.) Computer, Informatics, Cybernetics and Ap-
376 plications: CICA 2011. pp. 971–978. Springer Netherlands, Dordrecht (2012)
377 6. Itai, A., Papadimitriou, C.H., Szwarcfiter, J.L.: Hamilton paths in grid graphs.
378 SIAM Journal on Computing **11**(4), 676–686 (1982)
379 7. Ito, T., Demaine, E.D., Harvey, N.J., Papadimitriou, C.H., Sideri, M., Uehara, R.,
380 Uno, Y.: On the complexity of reconfiguration problems. Theoretical Computer
381 Science **412**(12), 1054 – 1065 (2011)
382 8. Jacobsen, J.L.: Exact enumeration of Hamiltonian circuits, walks and chains in
383 two and three dimensions. Journal of Physics A: Mathematical and General **40**,
384 14667–14678 (2007)
385 9. Keshavarz-Kohjerdi, F., Bagheri, A.: Hamiltonian paths in L-shaped grid graphs.
386 Theoretical Computer Science **621**, 37–56 (2016)
387 10. Lignos, I.: Reconfigurations of Combinatorial Problems: Graph Colouring and
388 Hamiltonian Cycle. Ph.D. thesis, Durham University (2017)
389 11. Muller, P., Hascoet, J.Y., Mognol, P.: Toolpaths for additive manufacturing of
390 functionally graded materials (FGM) parts. Rapid Prototyping Journal **20**(6), 511–
391 522 (2014)
392 12. Nishat, R.I.: Reconfiguration of Hamiltonian Cycles and Paths in Grid Graphs.
393 Ph.D. thesis, University of Victoria, Canada (2020)
394 13. Nishat, R.I., Whitesides, S.: Bend complexity and Hamiltonian cycles in grid
395 graphs. In: Cao, Y., Chen, J. (eds.) Computing and Combinatorics - 23rd Inter-
396 national Conference, COCOON 2017, Hong Kong, China, August 3-5, 2017, Pro-
397 ceedings. Lecture Notes in Computer Science, vol. 10392, pp. 445–456. Springer
398 (2017)
399 14. Nishat, R.I., Whitesides, S.: Reconfiguring Hamiltonian cycles in l-shaped grid
400 graphs. In: Sau, I., Thilikos, D.M. (eds.) Graph-Theoretic Concepts in Computer
401 Science - 45th International Workshop, WG 2019, Vall de Núria, Spain, June 19-21,
402 2019, Revised Papers. Lecture Notes in Computer Science, vol. 11789, pp. 325–337.
403 Springer (2019)
404 15. Nishimura, N.: Introduction to reconfiguration. Algorithms **11**(4), 52 (2018)
405 16. Bodroža Pantić, O., Pantić, B., Pantić, I., Bodroža Solarov, M.: Enumeration of
406 Hamiltonian cycles in some grid graphs. MATCH Communications in Mathemat-
407 ical and in Computer Chemistry **70** (01 2013)
408 17. Pettersson, V.: Enumerating Hamiltonian cycles. The Electronic Journal of Com-
409 binatorics **21**(4), P4.7 (2014)
410 18. Ruskey, F., Sawada, J.: Bent hamilton cycles in d-dimensional grid graphs. the
411 electronic journal of combinatorics **10**(1), R1 (2003)
412 19. Takaoka, A.: Complexity of Hamiltonian cycle reconfiguration. Algorithms **11**(9),
413 140 (2018)
414 20. Umans, C., Lenhart, W.: Hamiltonian cycles in solid grid graphs. In: 38th Ann.
415 Symp. on Foundations of Computer Science, FOCS '97. pp. 496–505 (1997)
416 21. Winter, S.: Modeling costs of turns in route planning. Geoinformatica **6**(4), 345–
417 361 (Dec 2002)