J Supercomput (2013) 65:723–741
DOI 10.1007/s11227-012-0852-0

An efficient parallel algorithm for the longest path problem in meshes

Fatemeh Keshavarz-Kohjerdi · Alireza Bagheri

Published online: 9 January 2013

Abstract The longest path problem is the problem of finding a simple path with the maximum number of vertices in a given graph, and so far it has been solved polynomially only for a few classes of graphs. This problem generalizes the well-known Hamiltonian path problem, hence it is NP-hard in general graphs. In this paper, first we give a sequential linear-time algorithm for the longest path problem in meshes. Then based on this algorithm, we present a constant-time parallel algorithm for the problem, which can be run on every parallel machine.

Keywords Grid graph · Longest path · Meshes · Sequential and parallel algorithms

1 Introduction

The longest path problem, i.e., the problem of finding a simple path with the maximum number of vertices, is one of the most important problems in graph theory. The well-known NP-complete Hamiltonian path problem, i.e., deciding whether there is a simple path that visits each vertex of the graph exactly once, is a special case of the longest path problem and has many applications [5, 7].

Only few polynomial-time algorithms are known for the longest path problem for special classes of graphs. This problem for trees began with the work of Dijkstra around 1960, and was followed by others [2, 9, 16, 18, 22]. In the area of approximation algorithms, it has been shown that the problem is not in APX, i.e., there is no polynomial-time constant factor approximation algorithm for the problem unless

Part of this research was conducted while F. Keshavarz-Kohjerdi was a M.Sc. student at Islamic Azad University, North Tehran Branch, Tehran, Iran.

F. Keshavarz-Kohjerdi (✉) · A. Bagheri
Department of Computer Engineering & IT, Amirkabir University of Technology, Tehran, Iran
e-mail: fatemeh.keshavarz@aut.ac.ir

P = NP [9]. Also, it has been shown that finding a path of length $n - n^{\epsilon}$ is not possible in polynomial-time unless P = NP [12]. For the background and some known results about approximation algorithms, we refer the reader to [1, 6, 24].

A grid graph is a graph in which vertices lie on integer coordinates and edges connect vertices that are separated by a distance of one. A solid grid graph is a grid graph without holes. A rectangular grid graph $R(m, n)$ is the subgraph of G^{∞} (the infinite grid graph) induced by $V(R) = \{\upsilon \mid 1 \leq \upsilon_x \leq m,\ 1 \leq \upsilon_y \leq n\}$, where υ_x and υ_y are respectively x and y coordinates of υ (see Fig. 1). A mesh $M(m, n)$ is a rectangular grid graph $R(m, n)$. Grid graphs can be useful representations in many applications. Myers [19] suggests modeling city blocks in which street intersection are vertices and streets are edges. He studied enumeration of tours in Hamiltonian rectangular latice graphs. Luccio and Mugnia [17] suggest using a grid graph to represent a two-dimensional array type memory accessed by a read/write head moving up, down or across. The vertices correspond to the center of each cell and edges connect adjacent cells. Finding a path in the grid corresponds to accessing all the data. They studied Hamiltonian paths on rectangular chessboards.

Itai et al. [11] have shown that the Hamiltonian path problem for general grid graphs, with or without specified endpoints, is NP-complete. The problem for rectangular grid graphs, however, is in P requiring only linear-time. Later, Chen et al. [3] improved the algorithm of [11] and presented a parallel algorithm for the problem in mesh architecture. Lenhart and Umans [15] have presented a polynomial-time algorithm for finding Hamiltonian cycles in solid grid graphs. Zamfirescu and Zamfirescu [23] have given sufficient conditions for a grid graph to be Hamiltonian and it is proved that all finite grid graphs of positive width have Hamiltonian line graphs.

Recently, Hamiltonian cycle (path) and longest path problems in grid graphs have received much attention. Salman [21] introduced a family of grid graphs, i.e., alphabet grid graphs, and determined classes of alphabet grid graphs that contain Hamiltonian cycles. Islam et al. [10] showed that the Hamiltonian cycle problem in hexagonal grid graphs is NP-complete. Gordon et al. [8] proved that all connected, and locally connected triangular grid graphs are Hamiltonian, and gave a sufficient condition for a connected graph to be fully cycle extendable and also showed that the Hamiltonian cycle problem for triangular grid graphs is NP-complete.

Zhang and Liu [25] gave an approximation algorithm for the longest path problem in grid graphs; their algorithm runs in quadratic time. Keshavarz-Kohjerdi et al. [13] studied the longest path problem for rectangular grid graphs; their algorithm is based on the divide and conquer technique and runs in linear time. Some other results about grid graphs are investigated in [14, 20].

In this paper, first we present a sequential and then a parallel algorithm for finding a longest path between two given vertices in a rectangular grid graph (mesh).

1.1 Our contribution

We present the first parallel algorithm for the longest path problem on meshes. This algorithm can be considered as an improvement and a parallel version of the algorithm in [13]. Our algorithm has improved the previous algorithm [13] by reducing the number of partition steps from $O(m + n)$ to a constant.

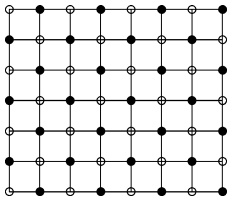


Fig. 1 The rectangular grid graph $R(8, 7)$

1.2 Organization of the paper

In Sect. 2, some necessary definitions and previous results are given. A sequential algorithm for the longest path problem is given in Sect. 3. In Sect. 4, a parallel algorithm for the problem is introduced, which is based on the sequential algorithm. The conclusion is given in Sect. 5.

2 Preliminary definitions and previous results

In this section, we give a few definitions and introduce the corresponding notations. We then gather some previously established results on the Hamiltonian and the longest path problems in grid graphs which have been presented in [3, 11, 13].

The *two-dimensional integer grid* G^{∞} is an infinite graph with vertex set of all the points of the Euclidean plane with integer coordinates. In this graph, there is an edge between any two vertices of unit distance. For a vertex v of this graph, let v_x and v_y denote x and y coordinates of its corresponding point, respectively (sometimes we use (v_x, v_y) instead of v). We color the vertices of the two-dimensional integer grid as black and white. A vertex v is colored *white* if $v_x + v_y$ is even, and it is colored *black* otherwise.

A *grid graph* G_g is a finite vertex-induced subgraph of the two-dimensional integer grid. In a grid graph G_g, each vertex has degree at most four. Clearly, there is no edge between any two vertices of the same color. Therefore, G_g is a bipartite graph. Note that any cycle or path in a bipartite graph alternates between black and white vertices. A *rectangular grid graph* $R(m, n)$ (or R for short) is a grid graph whose vertex set is $V(R) = \{v \mid 1 \leq v_x \leq m,\ 1 \leq v_y \leq n\}$. A mesh $M(m, n)$ is a rectangular grid graph $R(m, n)$; see Fig. 1.

In the figures, we assume that $(1, 1)$ is the coordinates of the vertex in the upper left corner. The size of $R(m, n)$ is defined to be mn. $R(m, n)$ is called *odd-sized* if mn is odd, and it is called *even-sized* otherwise.

Since even $\times$ odd rectangular grid graphs and odd $\times$ even rectangular grid graphs are isomorphic, so all rectangular grid graphs considered here are odd $\times$ odd, even $\times$ odd, and even $\times$ even. $R(m, n)$ is called a 1-rectangle if either $m = 1$ or $n = 1$, or a 2-rectangle if either $n = 2$ and $m > 1$, or $m = 2$ and $n > 1$.

The following lemma states a result about the Hamiltonicity of even-sized rectangular graphs.

Lemma 2.1 [3] *$R(m, n)$ has a Hamiltonian cycle if and only if it is even-sized and $m, n > 1$.*

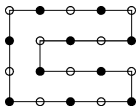


Fig. 2 A Hamiltonian cycle for the rectangular grid graph $R(5, 4)$

Figure 2 shows a Hamiltonian cycle for an even-sized rectangular grid graph, found by Lemma 2.1. Every Hamiltonian cycle found by this lemma contains all the boundary edges on the three sides of the rectangular grid graph. This shows that for an even-sized rectangular grid graph R, we can always find a Hamiltonian cycle, such that it contains all the boundary edges, except of exactly one side of R which contains an even number of vertices.

Two distinct vertices υ and υ' in $R(m, n)$ are called *color-compatible* if either both υ and υ' are white and $R(m, n)$ is odd-sized, or υ and υ' have different colors and $R(m, n)$ is even-sized. Let $(R(m, n), s, t)$ denote the rectangular grid graph $R(m, n)$ with two specified distinct vertices s and t. Without loss of generality, we assume $s_x \leq t_x$.

$(R(m, n), s, t)$ is called *Hamiltonian* if there exists a Hamiltonian path between s and t in $R(m, n)$. An even-sized rectangular grid graph contains the same number of black and white vertices. Hence, the two end-vertices of any Hamiltonian path in the graph must have different colors. Similarly, in an odd-sized rectangular grid graph the number of white vertices is one more than the number of black vertices. Therefore, the two end-vertices of any Hamiltonian path in such a graph must be white. Hence, the color-compatibility of s and t is a necessary condition for $(R(m, n), s, t)$ to be Hamiltonian. Furthermore, Itai et al. [11] showed that if one of the following conditions hold, then $(R(m, n), s, t)$ is not Hamiltonian:

(F1) $R(m, n)$ is a 1-rectangle and either s or t is not a corner vertex (Fig. 3(a)).
(F2) $R(m, n)$ is a 2-rectangle and (s, t) is a nonboundary edge, i.e., (s, t) is an edge and it is not on the outer face (Fig. 3(b)).
(F3) $R(m, n)$ is isomorphic to a 3-rectangle grid graph $R'(m, n)$ such that s and t is mapped to s' and t' and all of the following three conditions hold:

1. m is even,
2. s' is black, t' is white,
3. $s'_y = 2$ and $s'_x < t'_x$ (Fig. 3(c)) or $s'_y \neq 2$ and $s'_x < t'_x - 1$ (Fig. 3(d)).

They showed that $(R(m, n), s, t)$ is Hamiltonian if and only if s and t are color-compatible and $R(m, n)$, s and t do not satisfy any of conditions (F1), (F2), and (F3). In the following, we use $P(R(m, n), s, t)$ to indicate the problem of finding a longest path between vertices s and t in a rectangular grid graph $R(m, n)$, $L(R(m, n), s, t)$ to show the length of longest paths between s and t and $U(R(m, n), s, t)$ to indicate the upper bound on the length of longest paths between s and t. Keshavarz-Kohjerdi et al. [13] defined the following conditions:

(C0) s and t are color-compatible and none of (F1)–(F3) hold.
(C1) Neither (F1) nor (F2*) holds and either

1. $R(m, n)$ is even-sized and s and t are same-colored or
2. $R(m, n)$ is odd-sized and s and t are different-colored.

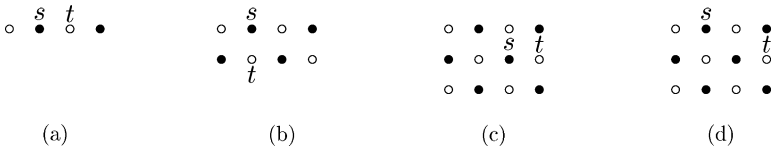


Fig. 3 The rectangular grid graphs in which there is no Hamiltonian path between s and t

(C2) 1. $R(m,n)$ is odd-sized and s and t are black-colored and neither (F1) nor (F2*) holds, or
2. s and t are color-compatible and (F3) holds,

where (F2*) is defined as follows:

(F2*) $R(m,n)$ is a 2-rectangle and $s_x = t_x$ or ($s_x = t_x - 1$ and $s_y \neq t_y$).

It is easy to show that any $(R(m,n),s,t)$ must satisfy one of conditions (C0), (C1), (C2), (F1) and (F2*). They proved the following upper bounds on the length of longest paths:

$$U\big(R(m,n),s,t\big) = \begin{cases} t_x - s_x + 1, & \text{if (F1)},\\ \max(t_x + s_x, 2m - t_x - s_x + 2), & \text{if (F2*)},\\ mn, & \text{if (C0)},\\ mn - 1, & \text{if (C1)},\\ mn - 2, & \text{if (C2)}. \end{cases}$$

Theorem 2.1 [13] *In a rectangular grid graph $R(m,n)$, a longest path between any two vertices s and t can be found in linear time and its length (i.e., $L(R(m,n),s,t)$) is equal to $U(R(m,n),s,t)$, which is defined before.*

3 The sequential algorithm

In this section, we present our sequential algorithm for finding a longest path between two given vertices in a rectangular grid graph. This algorithm is the base of our parallel algorithm which is introduced in Sect. 4. First, we solve the problem for 1-rectangles and 2-rectangles.

Lemma 3.1 [13] *Let $P(R(m,n),s,t)$ be a longest path problem with $n = 1$ or $n = 2$, then $L(R(m,n),s,t) = U(R(m,n),s,t)$.*

The proofs of lemmas are moved to the Appendix to improve readability of the paper.

Definition 3.1 [13] A *separation* of a rectangular grid graph R is a partition of R into two disjoint rectangular grid graphs R_1 and R_2, i.e., $V(R) = V(R_1) \cup V(R_2)$, and $V(R_1) \cap V(R_2) = \emptyset$.

Definition 3.2 [11] Let υ and υ' be two distinct vertices in R. If $\upsilon_x \leq 2$ and $\upsilon'_x \geq m-1$, then υ and υ' are called *antipodes*.

From now on, we assume that $m, n > 2$, so one of conditions (C0), (C1), and (C2) should hold. Following the technique used in [3], we develop an algorithm for finding longest paths.

Our sequential algorithm is given in Algorithm 3.1. In the following, we describe the steps of the algorithm in detail.

Algorithm 3.1 The sequential longest path algorithm

procedure SeqLongestPath($R(m,n), s, t$)
Input: A rectangular grid graph $R(m,n)$ with two disjoint vertices s and t
Output: A longest path of G
Step 1. By a peeling operation, partitions $R(n,m)$ into five disjoint rectangular grid subgraphs R_1 to R_5, such that $s, t \in R_5$
Step 2. Find a longest path between s and t in R_5
Step 3. Construct Hamiltonian cycles in rectangular grid subgraphs R_1 to R_4
Step 4. Construct a longest path between s and t in R by combining the Hamiltonian cycles of R_1 to R_4 and the longest path of R_5

3.1 Peeling operation

In this section, we describe Step 1 of Algorithm 3.1. In this step, $R(m,n)$ is partitioned into disjoint rectangular grid subgraphs.

Definition 3.3 [3] Partitioning a rectangular grid graph R into five disjoint rectangular grid subgraphs R_1–R_5 that is done by two horizontal and two vertical separations are called *peeling operation*, if the following two conditions hold:

1. $s, t \in R_5$, and s and t are antipodes.
2. Four rectangular grid subgraphs R_1–R_4 are even-sized rectangular grid graphs whose two boundary sizes are both greater than one, or are null rectangles.

Generally the two vertical separation of a peeling are done before the two horizontal separation. However, for an odd $\times$ odd rectangular grid graph with $s_x = t_x$, this order is reversed to guarantee that the boundary sizes of R_3 and R_4 are greater than one. Figure 4 shows a peeling operation on $R(15,11)$ where s is $(6,5)$ and t is $(8,9)$, and on $R(9,9)$ where s is $(5,3)$ and t is $(5,6)$.

The following lemma can be obtained directly from Definition 3.3.

Lemma 3.2 [3] *Let $R_5(n_5, m_5)$ be the resulting rectangular grid subgraph of a peeling operation on $R(n,m)$, where $s, t \in V(R_5)$. Then*

1. *s and t keep the same colors in R_5 as in R; and*
2. *R_5 has the same parity as R, that is, $m_5 \bmod 2 = m \bmod 2$, and $n_5 \bmod 2 = n \bmod 2$.*

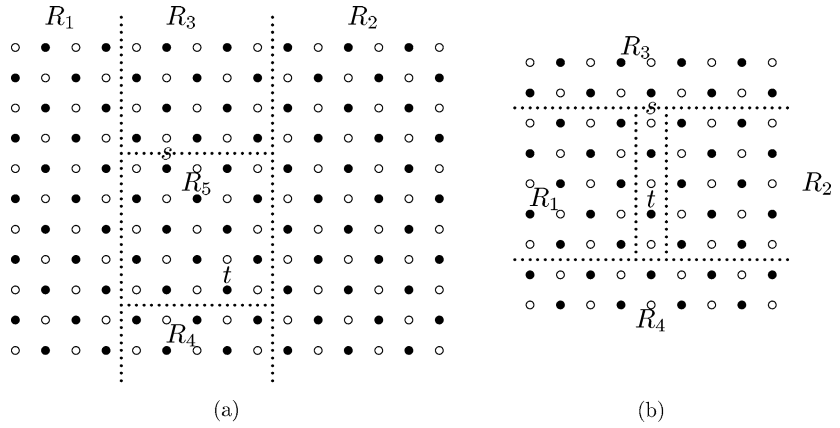


Fig. 4 A peeling operation on $R(15, 11)$ and $R(9, 9)$

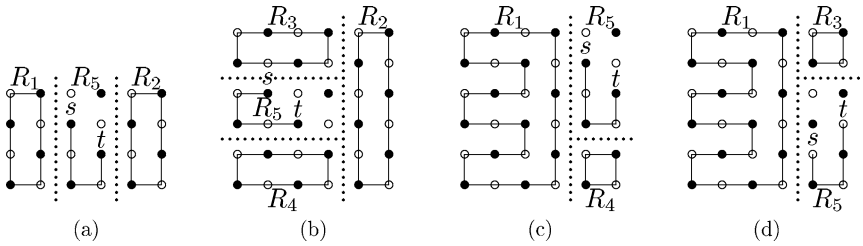


Fig. 5 Rectangular grid graphs in which a peeling operation is not proper

Definition 3.4 A peeling operation on R is called *proper* if $|R_1| + |R_2| + |R_3| + |R_4| + U(R_5, s, t) = U(R(n, m), s, t)$, where $|R_i|$ denotes the number of vertices of R_i.

Lemma 3.3 *For the longest path problem $P(R(n, m), s, t)$, any peeling operation on $R(m, n)$ is proper if either*:

1. *Condition* (*C*0) *holds and* $m \bmod 2 = n \bmod 2$ (*i.e.* $R(m, n)$ *is* even $\times$ even *or* odd $\times$ odd), *or*
2. *One of conditions* (*C*1) *and* (*C*2) *hold and* $R(m, n)$ *is* even $\times$ odd *or* odd $\times$ odd.

Nevertheless, a peeling operation on an even $\times$ even rectangular grid graph $R(m, n)$ may not be proper, see Fig. 5 where the dotted lines represent a peeling operation. In the two following cases, a peeling operation is not proper:

(F1′) s is black, $t_y = s_y + 1$ and $s_x \neq t_x$.
(F2′) s is white, $t_y = s_y - 1$ and $s_x \neq t_x$.

Lemma 3.4 *For the longest path problem $P(R(n, m), s, t)$, where $R(m, n)$ is an* even $\times$ even *rectangular grid graph, a peeling operation on $R(m, n)$ is proper if and only if $P(R(n, m), s, t)$ is not in cases* (*F*1′) *and* (*F*2′).

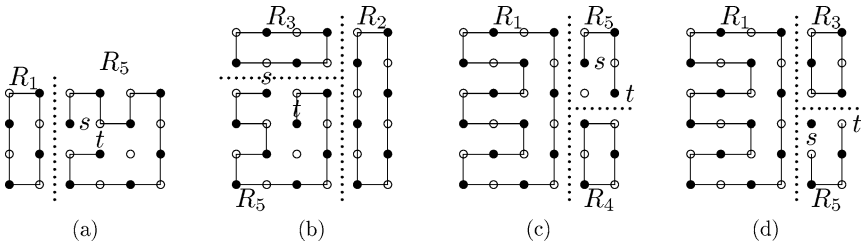


Fig. 6 Rectangular grid graphs in which a peeling operation can be made proper by adjusting the peeling boundaries

When a peeling operation is not proper it can be made proper by adjusting the peeling boundaries. In that case, if R_1, R_2, R_3, and R_4 are empty, then R_5 is 2-rectangle that is in case (F2*). Therefore, without loss of generality, we assume R_1, R_2, R_3, or R_4 is not empty. If R_1 or R_2 is not empty, then we move one column (or two columns when R_1 or R_2 is a 2-rectangle) from R_1 or R_2 to R_5 such that R_1 and R_2 are still even-sized rectangular grid graphs; see Fig. 6(a). If R_3 or R_4 is not empty, then we move one row (or two rows when R_3 or R_4 is 2-rectangle) from R_3 or R_4 to R_5 (Fig. 6(b)), or move the bottom row from R_5 to R_4 (Fig. 6(c)) or move the upper row from R_5 to R_3 (Fig. 6(d)), such that R_3 and R_4 are still even-sized rectangular grid graphs.

3.2 Finding a longest path in $R_5(m_5, n_5)$

In this section, we present Step 2 of Algorithm 3.1. Let $R(m, n)$ be an rectangular grid graph and let $R_5(m_5, n_5)$ be R_5 subrectangle of $R(m, n)$ constructed in Step 1. Now, we construct a longest path in $R_5(m_5, n_5)$. The following cases can be considered for $R_5(m_5, n_5)$:

(a) $m_5, n_5 \leq 3$.
(b) m_5, n_5 are even, and either $m_5 \geq 4$ or $n_5 \geq 4$;
(c) m_5, n_5 are odd, and either $m_5 \geq 5$ or $n_5 \geq 5$;
(d) m_5 is even and n_5 is odd, and either $m_5 \geq 4$ or $n_5 \geq 5$.

For case (a), we showed that when $n = 1, 2$ the problem can be solved easily. For $m, n = 3$ the longest paths of all the possible problems are depicted in Fig. 7 (the isomorphic cases are omitted). For cases (b), (c), and (d), we use the definition of trisecting.

Definition 3.5 [3] Two separations of R_5 that partition it into three rectangular grid subgraphs R_5^s, R_5^t and R_5^m is called *trisecting* (see Fig. 8), if

(i) R_5^s and R_5^t are 2-rectangles, and
(ii) $s \in V(R_5^s)$ and $t \in V(R_5^t)$.

A trisecting can be done by two ways horizontally and vertically. If $n_5 \geq 4$, then trisecting is done horizontally, otherwise trisecting is done vertically.

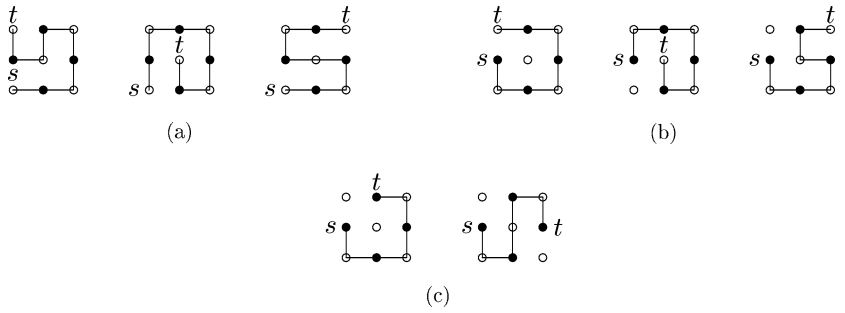
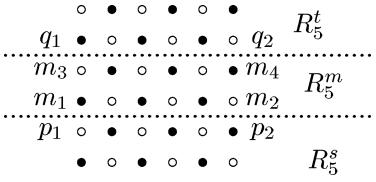


Fig. 7 For $n = m = 3$, (**a**) s and t are *white*, then there is Hamiltonian path, (**b**) s and t have different colors, then there is a path with $U(R, s, t) = mn - 1$ and (**c**) s and t are *black*, then there is a path with $U(R, s, t) = mn - 2$

Fig. 8 A trisecting on $R(6, 6)$



Definition 3.6 The corner vertices p and q respectively on the boundary of R_5^s and R_5^t facing R_5^m is called *junction vertices* if either

(i) Condition (C0) holds and they have different colors from s and t, or
(ii) One of conditions (C1) and (C2) holds and $U(R_5^s, s, p) + U(R_5^m, m, m') + U(R_5^t, q, t) = U(R_5(m, n), s, t)$. where m and m' are two of the corner vertices of R_5^m facing R_5^s and R_5^t.

In Fig. 8, p_1 and p_2, q_1 and q_2, m_1, m_2, m_3 and m_4 are junction vertices in R_5^s, R_5^t, and R_5^m, respectively. Existence of junction vertices has been proved for condition (C0) in [3]; in this paper, we only consider conditions (C1) and (C2).

Lemma 3.5 *Performing a trisecting on R_5, where $m, n > 3$, and assuming condition (C1) or (C2) holds, if $n_5 = 4$, and s and t are on the common border of R_5^s and R_5^t (R_5^m is null), then there is no junction vertex for R_5^s and R_5^t, otherwise R_5^s and R_5^t have at least one junction vertex.*

After trisecting, we construct a longest path or Hamiltonian path in R_5^s, R_5^m and R_5^t between s and p, m and m', and q and t, respectively. In the case that none of R_5^s and R_5^t have junction vertices (when $n_5 = 4$ and both s and t are on the common border of R_5^s and R_5^t), we construct a longest path in R_5^s (resp. R_5^t) between s (resp. t) and a noncorner vertex of the boundary facing R_5^t (resp. R_5^s); see Fig. 9(a). At the end, the paths in R_5 are combined through the junction vertices to make a simple longest path in R_5; see Figs. 9(b), 9(c).

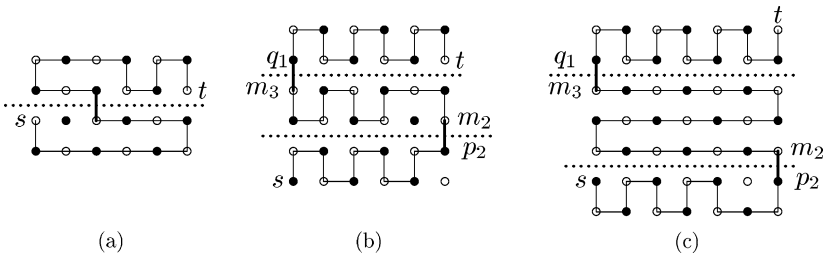


Fig. 9 The Longest path in $R(6,4)$, $R(6,6)$, and $R(7,7)$

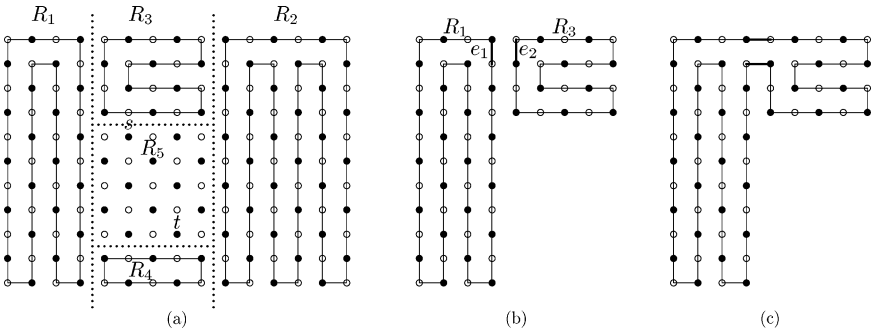


Fig. 10 (**a**) Hamiltonian cycles in R_1 to R_4, (**b**) and (**c**) combining two Hamiltonian cycles

3.3 Construct Hamiltonian cycles in rectangular grid subgraphs and then a longest path in $R(m,n)$

In this section, we present Steps 3 and 4 of Algorithm 3.1. We construct Hamiltonian cycles in rectangular grid subgraphs R_1 to R_4 by Lemma 2.1; see Fig. 10(a), and merge all Hamiltonian cycles to a single Hamiltonian cycle.

Two nonincident edges e_1 and e_2 are parallel, if each end vertex of e_1 is adjacent to an end vertex of e_2. Using two parallel edges e_1 and e_2 of two Hamiltonian cycles, such as two darkened edges of Fig. 10(b), we can combine the cycles as illustrated in Fig. 10(c), and obtain a larger cycle.

Now, we describe combining a longest path in R_5 with the constructed cycle. Without loss of generality, suppose that at least one of R_1, R_2, R_3, and R_4, say R_i, exists otherwise there is nothing to be merged with R_5. Note that the Hamiltonian cycle in R_i is constructed such that it contains all the boundary edges facing R_5.

Furthermore, any path P of length $U(R_5,s,t)$ in R_5 contains all the vertices of R_5 except one or two vertices as mentioned before. Therefore, P should contain a boundary edge of R_5 that has a parallel edge in R_i (see Fig. 11), except when R_5 is a 2-rectangle, in this case R_5 may have no boundary edge parallel to any edge of R_i (see Figs. 12(a), 12(d)). But in this case, s should be adjacent to R_i. Let an edge (s,v) of R_5 be adjacent to R_i. Then P must contain an edge (s,u) such that u is not adjacent to R_i as depicted in Figs. 12(a), 12(d) (we may need to swap the roles of s and t to find such edges). Hence, replacing the role of u with v we can modify P

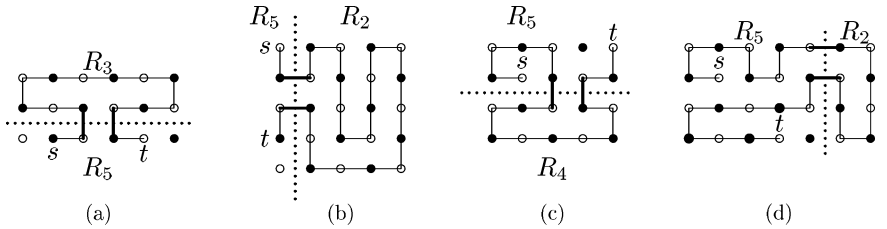


Fig. 11 Combining Hamiltonian cycle with a longest path in R_5

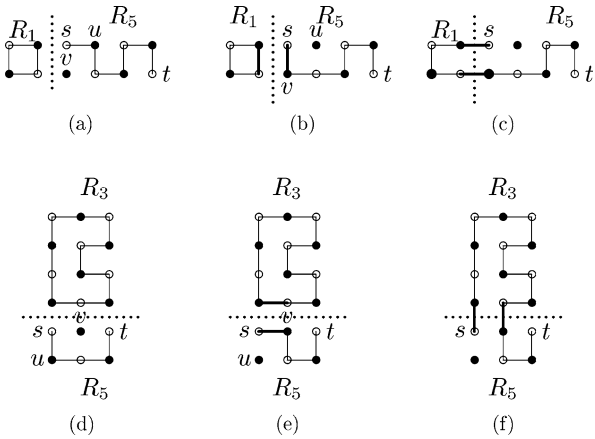


Fig. 12 Combining Hamiltonian cycle with a longest path in R_5, when R_5 is a 2-rectangle and s and t are the corners of R_5

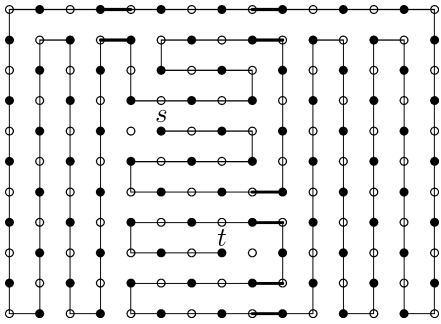


Fig. 13 A longest path between s and t

such that it contains a boundary edge adjacent to R_i (see Figs. 12(b), 12(e)). Using two parallel edges of P and the Hamiltonian cycle of R_i such as the two darkened edges of Figs. 12(b), 12(e) we can combine them as illustrated in Figs. 12(c), 12(f). Combining the Hamiltonian cycles of R_1–R_4 with the longest path of R_5 results in a longest path of $R(m, n)$; see Fig. 13.

3.4 The time complexity of algorithm

Consider the pseudo-code of our algorithm in Algorithm 3.1. Step 1 does only a constant number of partitioning, during the peeling operation, which is done in constant

time. Step 2 trisects R_5, which requires also a constant number of partitioning operations and finds a longest path in R_5 by merging paths of the partitions which can be done in linear time. Step 3 finds Hamiltonian cycles of R_1 to R_4 which is done in linear time. Step 4 which combines the Hamiltonian cycles and the longest path requires only constant time. Therefore, in total, our sequential algorithm has linear time complexity.

4 The parallel algorithm

In this section, we present a parallel algorithm for the longest path problem. This algorithm is based on the sequential algorithm presented in the previous section. Our parallel algorithm runs on every parallel machine, we do not need any interprocessor communication in our algorithm. We assume there are nm processors and they work in SPMD mode. For simplicity, we use a two-dimensional indexing scheme. Each vertex v of the given rectangular grid graph $R(m, n)$ is mapped to processor (v_x, v_y). Each processor knows its index, coordinates s and t, and m and n.

The peeling phase is parallelized easily, every processor calculates the following four variables, in parallel [3]:

$$r_1 = \begin{cases} s_x - 2; & s_x \bmod 2 = 0 \\ s_x - 1; & \text{otherwise} \end{cases}$$

$$r_2 = \begin{cases} t_x + 1; & t_x \bmod 2 = m \bmod 2 \\ t_x + 2; & \text{otherwise} \end{cases}$$

$$r_3 = \begin{cases} \min(s_y, t_y) - 2; & \min(s_y, t_y) \bmod 2 = 0 \\ \min(s_y, t_y) - 1; & \text{otherwise} \end{cases}$$

$$r_4 = \begin{cases} \max(s_y, t_y) + 1; & \max(s_y, t_y) \bmod 2 = n \bmod 2 \\ \max(s_y, t_y) + 2; & \text{otherwise} \end{cases}$$

where variables r_1, r_2, r_3, and r_4 correspond to the right-most column number of R_1, the left-most column number of R_2, the bottom row number of R_3, and the top row number of R_4, respectively. Then a processor can identify its subrectangle by comparing its coordinates with these four variables. In cases (F1′) and (F2′), the boundary adjustment can be done by simply decrementing R_1, R_2, R_3, or R_4 or incrementing R_3 or R_4.

The trisecting phase is also parallelized in a similar manner. In the following, we describe how we parallelized the horizontal trisecting, in two cases $R(m, n)$ is even × odd (or odd × odd) and it is even × even. In the case $R(m, n)$ is even × odd or odd × odd, every processor simultaneously calculates the following two variables:

$$b = \begin{cases} \min(s_y, t_y) & \min(s_y, t_y) \bmod 2 = 0 \\ \min(s_y, t_y) + 1 & \min(s_y, t_y) \bmod 2 \neq 0 \end{cases}$$

$$u = \begin{cases} \max(s_y, t_y) & \max(s_y, t_y) \bmod 2 = 0 \\ \max(s_y, t_y) - 1 & \max(s_y, t_y) \bmod 2 \neq 0 \end{cases}$$

where variables b and u correspond to the bottom row number of R_5^s (resp. R_5^t), and the top row number of R_5^t (resp. R_5^s), respectively.

In the case $R(m, n)$ is even $\times$ even, every processor simultaneously calculates the following two variables:

$$b = \begin{cases} \min(s_y, t_y) & \min(s_y, t_y) \bmod 2 = 0 \\ \min(s_y, t_y) + 1 & \min(s_y, t_y) \bmod 2 \neq 0 \end{cases}$$

$$u = \begin{cases} \max(s_y, t_y) & \max(s_y, t_y) \bmod 2 \neq 0 \\ \max(s_y, t_y) - 1 & \max(s_y, t_y) \bmod 2 = 0 \end{cases}$$

A similar method can be used to parallelize the vertical trisecting.

After peeling and trisecting, all processors in the same subrectangles simultaneously construct either a longest path, Hamiltonian path or cycle according to the pattern associated with the subrectangle. For constructing a Hamiltonian path in a rectangular grid graph, we use the constant-time algorithm of [3]. For constructing a Hamiltonian cycle in an even-sized rectangle, we use the constant-time algorithm of [4] in which every processor computes its successor in the cycle. Such an algorithm is given in Algorithm 4.1; see Fig. 14(a).

For constructing a longest path between s and t in R_5, since R_5^s and R_5^t are 2-rectangles, then a Hamiltonian or longest path can be easily constructed using Lemma 3.1. For R_5^m, there are two cases:

Case 1. R_5^m is odd sized. Since four vertices m_1, m_2, m_3, and m_4 are white, then we construct Hamiltonian path from m_1 or m_2 to m_3 or m_4 by algorithm of [3].

Case 2. R_5^m is even sized. In this case, Figs. 14(b), 14(c) show that different patterns for constructing a longest subpath between vertices (m, n) and $(m, 1)$ by Algorithm 4.2, and between $(1, n)$ and $(m, 1)$ by Algorithm 4.3, respectively. The algorithms for other patterns can be derived in a similar way.

Then the combining phase is parallelized as follows. The two processors at the two endpoints of an edge e_1 in a Hamiltonian cycle c_1 make sure whether an adjacent Hamiltonian cycle c_2 exists or not. Then their successors are modified to the adjacent processors in c_2 if c_2 exists. In the same way, the two processors at the endpoints of an edge in the longest path P in R_5 as well make sure the existence of an adjacent edge in Hamiltonian cycle C, and modify their successors. Therefore, the combining phase can be parallelized in constant steps with no inter-processor communication.

5 Conclusion and future work

We presented a linear-time sequential algorithm for finding a longest path in a rectangular grid graph between any two given vertices. Based on the sequential algorithm, a constant-time parallel algorithm is introduced for the problem, which can be run on every parallel machine.

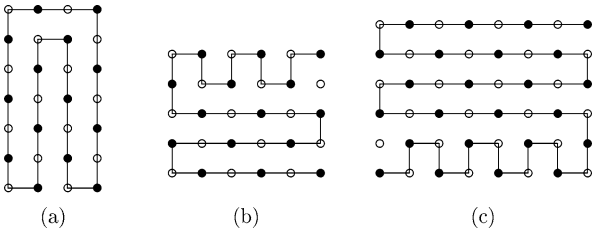


Fig. 14 (**a**) A Hamiltonian cycle in $R(4, 7)$, (**b**) and (**c**) two patterns of longest path in $R_5(5, 5)$ and $R_5(8, 6)$

Algorithm 4.1 The parallel Hamiltonian cycle algorithm for an even-sized rectangular grid graphs

procedure HAM-CYCLE$R(m, n)$

1: **for** each processor (x, y) in $R(m, n)$ **do** in parallel
2: **if** $y = 1$ and $x < m$, **then** successor $(x, y) \leftarrow (x + 1, y)$
3: **elseif** ($y = 2$, x is odd and $x \neq 1$) or ($y = n$ and x even), **then** successor $(x, y) \longleftarrow (x - 1, y)$
4: **elesif** x is even and $y < n$, **then** successor $(x, y) \longleftarrow (x, y + 1)$
5: **elseif** x is odd and $y > 2$, **then** successor $(x, y) \longleftarrow (x, y - 1)$

Algorithm 4.2 The parallel longest subpath algorithm for R_5^m (Fig. 14(b))

procedure LongestsubPath(R_5^m, p, q)

1: **for** each processor (x, y) in $R_5(m_5, n_5)$ **do** in parallel
2: **if** ($x = m$ and $y = 2$), **then** successor $(x, y) \longleftarrow$ nill
3: **elseif** y is odd and $x > 1$, **then** successor $(x, y) \longleftarrow (x - 1, y)$
4: **elesif** (y is odd and $x = 1$) or ($y = 2$ and x is odd) or (y is even and $x = m$), **then** successor $(x, y) \longleftarrow (x, y - 1)$
5: **elseif** $y = 1$ and x is even, **then** successor $(x, y) \longleftarrow (x, y + 1)$
6: **elseif** (y is even and $x < m$) or ($y = 2$ and x is even) or ($y = 1$ and x is odd), **then** successor $(x, y) \longleftarrow (x + 1, y)$

Algorithm 4.3 The parallel longest subpath algorithm for R_5^m (Fig. 14(c))

procedure LongestsubPath (R_5^m, p, q)

1: **for** each processor (x, y) in $R_5(m_5, n_5)$ **do** in parallel
2: **if** $x = 1$ and $y = n - 1$, **then** successor $(x, y) \longleftarrow$ nill
3: **elseif** (y is odd and $x = m$) or (y is even and $x = 1$) or ($y = n$ and x is even) **then** successor $(x, y) \longleftarrow (x, y - 1)$
4: **elesif** (y is odd and $x < m$) or ($y = n - 1$ and x is even) or ($y = n$ and x is odd) **then** successor $(x, y) \longleftarrow (x + 1, y)$
5: **elseif** y is even and $x > 1$ **then** successor $(x, y) \longleftarrow (x - 1, y)$
6: **elseif** $y = n - 1$ and x is odd **then** successor $(x, y) \longleftarrow (x, y + 1)$

The algorithm initially divides $R(m, n)$ into five subrectangles, then builds Hamiltonian cycles in the four subrectangles and a Hamiltonian path or longest path in the other subrectangle. Next, all Hamiltonian cycles are combined to a single Hamilto-

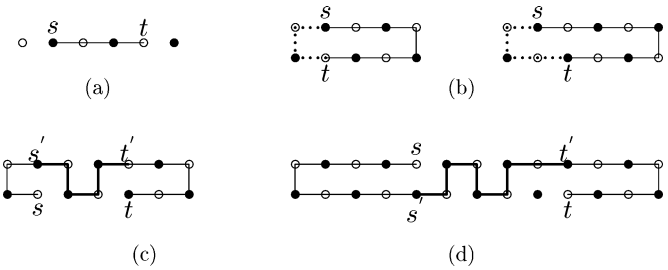


Fig. 15 (**a**) The longest path between s and t in a 1-rectangle, (**b**) The longest path between s and t in a 2-rectangle, (**c**) and (**d**) Longest paths with length $2m$ and $2m-1$ for a 2-rectangle, respectively

nian cycle. Finally, it combines all the paths and the cycle to a longest path. Our algorithm is a parallel version of the previous sequential algorithm [13], and is an improvement of it by reducing the number of partitioning steps from $O(m+n)$ to a constant.

Since the longest path problem is NP-hard in general grid graphs [11], it remains open if the problem is polynomially solvable in solid grid graphs. Further study can be done on the Hamiltonian (or longest) path problem in other special classes of graphs.

Appendix

Lemma 6.1 [13] *Let $P(R(m,n),s,t)$ be a longest path problem with $n=1$ or $n=2$, then $L(R(m,n),s,t)=U(R(m,n),s,t)$.*

Proof For a 1-rectangle obviously the lemma holds for the single possible path between s and t (see Fig. 15(a)). For a 2-rectangle, if removing s and t splits the graph into two components, then the path going through all vertices of the larger component has the length equal to $U(R(m,n),s,t)$ (see Fig. 4(b)). Otherwise, let s' be the vertex adjacent to s and t' be the vertex adjacent to t such that $s'_y \neq s_y$ and $t'_y \neq t_y$. Then we make a path from s to s' and a path from t to t' as shown in Figs. 15(c), 15(d), and connect s' to t' by a path such that at most one vertex remains out of the path as depicted in this figure. □

Lemma 6.2 *For the longest path problem $P(R(n,m),s,t)$, any peeling on $R(m,n)$ is proper if either*:

1. *Condition* (*C*0) *holds and* $m \bmod 2 = n \bmod 2$ (*i.e.*, $R(m,n)$ *is* even × even *or* odd × odd), *or*
2. *One of conditions* (*C*1) *and* (*C*2) *hold and* $R(m,n)$ *is* even × odd *or* odd × odd.

Proof The lemma has been proved for the case that (C0) holds (see [3]). So, we consider conditions (C1) and (C2). From Lemma 3.2, we know that s and t are still

color-compatible, and we are going to prove that $P(R_5(m_5, n_5), s, t)$ is not in cases $F1$ and $F2^*$.

By Lemma 3.1, when $R(m, n)$ is an odd $\times$ odd rectangular grid graph, $R_5(m_5, n_5)$ is also an odd $\times$ odd rectangular grid graph, s and t have the same color as in R, and hence $R_5(m_5, n_5)$ is not a 2-rectangle. If R_5 is a 1-rectangle, then $s_y = t_y$ or $s_x = t_x$ and then we have the two following cases:

Case 1. (C1) holds, and s and t have different colors. In this case, one of s_x and t_x (s_y and t_y) is even and the other is odd. Considering that s and t are antipodes and R_5 is odd $\times$ odd, one of s and t must be at the corner, and exactly one vertex goes out of the path.

Case 2. (C2) holds and both s and t are black. In this case, s_x, s_y, t_x and t_y are even. Hence, vertices s and t are before corner vertices and exactly two vertices go out of the path.

In a similar way, when $R(m, n)$ is an even $\times$ odd rectangular grid graph ((C1) holds), $R_5(m_5, n_5)$ is also an even $\times$ odd rectangular grid graph, and hence $R_5(m_5, n_5)$ is not a 2-rectangle. If R_5 is a 1-rectangle, then $s_y = t_y$. In this case, s_x and t_x are both odd or even. Hence, s or t are at the corner and exactly one vertex goes out of the path.

Therefore by Theorem 2.1, $U(R_5(m_5, n_5), s, t) = U(R(m, n), s, t)$ and the peeling of $R(m, n)$ is proper. □

Lemma 6.3 *For the longest path problem $P(R(n, m), s, t)$, where $R(m, n)$ is an* even $\times$ even *rectangular grid graph, a peeling operation on $R(m, n)$ is proper if and only if $P(R(n, m), s, t)$ is not in cases $(F1')$ and $(F2')$.*

Proof The proof is a simple by case analysis. □

Lemma 6.4 *Performing a trisecting on R_5, where $m, n > 3$, and assuming condition (C1) or (C2) holds, if $n_5 = 4$, and s and t are on the common border of R_5^s and R_5^t (R_5^m is null), then there is no junction vertex for R_5^s and R_5^t, otherwise R_5^s and R_5^t have at least one junction vertex.*

Proof Consider Figs. 16(a) and 16(b), where $n_5 = 4$ and two vertices s and t are on the common border of R_5^s and R_5^t. In this case, the only two vertices p_2 and q_1 may be junction vertices. By Theorem 2.1, there exists a Hamiltonian path from s to p_2 and from q_1 to t in R_5^s and R_5^t, respectively, and $U(R_5^s, s, p_2) + U(R_5^t, q_1, t) \neq U(R_5(m, n), s, t)$. Hence, neither R_5^s nor R_5^t has a junction vertex. Now for the other cases, we show that R_5^s and R_5^t have at least one junction vertex.

In case (b), s and t have the same colors (white or black), and two corner vertices on the boundary of R_5^s (resp., R_5^t) facing R_5^m have different colors and also R_5^m is a k-rectangle, where k is zero or $k \geq 2$ and even. We consider the following three cases for s and t:

Case 1. Both s and t are the corner vertices on the boundary of R_5^s and R_5^t facing R_5^m; see Fig. 16(c). By Theorem 2.1, there exists a Hamiltonian path from s to p_2 and from q_1 to t in R_5^s and R_5^t, respectively, and a path from m_3 to m_2 which does not contain a vertex in R_5^m. Therefore, $U(R_5^s, s, p_2) + U(R_5^m, m_3, m_2) + U(R_5^t, q_1, t) = U(R_5(m, n), s, t)$, and hence both R_5^s and R_5^t have a unique junction vertex.

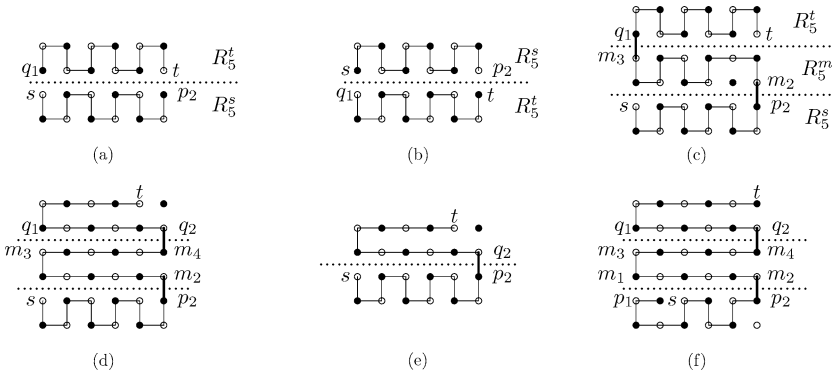


Fig. 16 A trisecting on $R(6, 6)$ and $R(6, 4)$

Case 2. Only s is the corner vertex on the boundary of R_5^s facing R_5^m; see Fig. 16(d). By Theorem 2.1, there exists a Hamiltonian path from s to p_2, from q_1 to t and a path from m_3 to m_2 which does not contain one vertex, or a Hamiltonian path from s to p_2, from m_2 to m_4 and a path from q_2 to t, which does not contain one vertex. Therefore, $U(R_5^s, s, p_2) + U(R_5^m, m_3, m_2) + U(R_5^t, q_1, t) = U(R_5(m, n), s, t)$ or $U(R_5^s, s, p_2) + U(R_5^m, m_2, m_4) + U(R_5^t, q_2, t) = U(R_5(m, n), s, t)$ and hence R_5^s has a unique junction vertex and R_5^t have two junction vertices (the same argument is also applied to t). In this case, where $n_5 = 4$, both R_5^s and R_5^t have a unique junction vertex; see Fig. 16(e).

Case 3. Neither s nor t are corner vertices on the boundary of R_5^s and R_5^t facing R_5^m; see Fig. 16(f). By Theorem 2.1, there exists a Hamiltonian path from s to p_1, from m_1 to m_3 and a path from q_1 to t which does not contain a vertex, or a Hamiltonian path from s to p_1, from q_2 to t and a path from m_1 to m_4 which does not contain a vertex, or a Hamiltonian path form m_2 to m_4, from q_2 to t and a path from s to p_2 which does not contain a vertex. Therefore, $U(R_5^s, s, p_1) + U(R_5^m, m_1, m_3) + U(R_5^t, q_1, t) = U(R_5(m, n), s, t)$, $U(R_5^s, s, p_1) + U(R_5^m, m_1, m_4) + U(R_5^t, q_2, t) = U(R_5(m, n), s, t)$ or $U(R_5^s, s, p_2) + U(R_5^m, m_2, m_4) + U(R_5^t, q_2, t) = U(R_5(m, n), s, t)$ and hence both R_5^s and R_5^t have two junction vertices.

In case (c), vertices s and t are black or have different colors, and the two corner vertices on the boundary of R_5^s (resp., R_5^t) facing R_5^m are black and also R_5^m is a k-rectangle, where $k \geq 1$ and odd. There are three cases for s and t:

Case 1. Both s and t are the corner vertices on the boundary of R_5^s and R_5^t facing R_5^m; see Fig. 17(a). Then s and t are black. By Theorem 2.1, there exists a path from s to p_2 in R_5^s, and a path from q_1 to t in R_5^t, which do not contain a vertex, and a Hamiltonian path from m_3 to m_2 in R_5^m. Therefore, $U(R_5^s, s, p_2) + U(R_5^m, m_3, m_2) + U(R_5^t, q_1, t) = U(R_5(m, n), s, t)$ and hence both R_5^s and R_5^t have a unique junction vertex.

Case 2. Only s is the corner vertex on the boundary of R_5^s facing R_5^m, then s is black and t is black or white; see Fig. 17(b). By Theorem 2.1, there exists a path from s to p_2; and a path from q_1 (or q_2) to t, where t is black, which does not contain a vertex, and a Hamiltonian path from and m_2 to m_4 (or from m_3

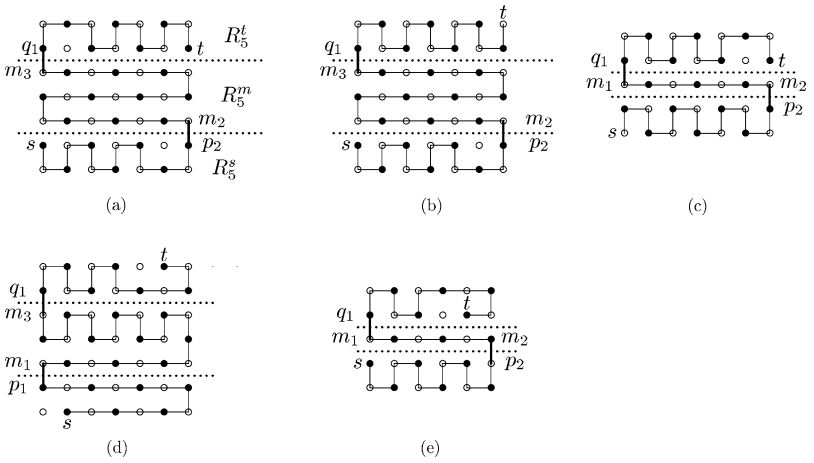


Fig. 17 A trisecting on $R(7,7)$, $R(7,5)$, and $R(6,5)$

to m_2), or a Hamiltonian path from q_1 (or q_2) to t, where t is white and m_2 to m_4 (or from m_3 to m_2) and a path from s to p_2, which does not contain a vertex. Therefore, $U(R_5^s, s, p_2) + U(R_5^m, m_3, m_2) + U(R_5^t, q_1, t) = U(R_5(m,n), s, t)$ or $U(R_5^s, s, p_2) + U(R_5^m, m_2, m_4) + U(R_5^t, q_2, t) = U(R_5(m,n), s, t)$ and hence R_5^s has a unique junction vertex and R_5^t have two junction vertices (the same argument is also applied to t). In this case, where $n_5 = 5$, both R_5^s and R_5^t have a unique junction vertex; see Fig. 17(c).

Case 3. Neither s nor t are corner vertices on the boundary of R_5^s and R_5^t facing R_5^m; see Fig. 17(d). By Theorem 2.1, there exists a Hamiltonian path from s to p, m to m' and q to t, where s (or t) is white, and a path from s to p and a path from q to t do not contain a vertex where s (or t) is black, p is p_1 or p_2, q is q_1 or q_2, m is m_1 or m_2 and m' is m_3 or m_4. Therefore, $U(R_5^s, s, p) + U(R_5^m, m, m') + U(R_5^t, q, t) = U(R_5(m,n), s, t)$, and hence both R_5^s and R_5^t have two junction vertices.

In case (d), if $n_5 > 3$, the trisecting is performed horizontally, and the claim is proved by applying the same argument for case (b); see Fig. 17(e). If $n_5 = 3$, the trisecting is performed vertically and also two corner vertices on the boundary of R_5^s facing R_5^m are black. Therefore, the claim is proved by applying the same argument for case (c). □

References

1. Björklund A, Husfeldt T (2003) Finding a path of superlogarithmic length. SIAM J Comput 32(6):1395–1402
2. Bulterman RW, van der Sommen FW, Zwaan G, Verhoeff T, van Gasteren AJM, Feijen WHJ (2002) On computing a longest path in a tree. Inf Process Lett 81(2):93–96
3. Chen SD, Shen H, Topor R (2002) An efficient algorithm for constructing Hamiltonian paths in meshes. Parallel Comput 28(9):1293–1305
4. Chen SD, Shen H, Topor RW (1995) Efficient parallel permutation-based range-join algorithms on meshconnected computers. In: Proceedings of the 1995 Asian computing science conference, Pathumthani, Thailand. Springer, Berlin, pp 225–238

5. Diestel R (2000) Graph theory. Springer, New York
6. Gabow HN, Nie S (2008) Finding long paths, cycles and circuits. In: 19th annual international symp on algorithms and computation (ISAAC). Lecture Notes in Computer Science, vol 5369, pp 752–763
7. Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. Freeman, San Francisco
8. Gordon VS, Orlovich YL, Werner F (2008) Hamiltonian properties of triangular grid graphs. Discrete Math 308:6166–6188
9. Gutin G (1993) Finding a longest path in a complete multipartite digraph. SIAM J Discrete Math 6(2):270–273
10. Islam K, Meijer H, Nunez Y, Rappaport D, Xiao H (2007) Hamiltonian circuts in hexagonal grid graphs. In: CCCG, pp 20–22
11. Itai A, Papadimitriou C, Szwarcfiter J (1982) Hamiltonian paths in grid graphs. SIAM J Comput 11(4):676–686
12. Karger D, Montwani R, Ramkumar GDS (1997) On approximating the longest path in a graph. Algorithmica 18(1):82–98
13. Keshavarz-Kohjerdi F, Bagheri A, Asgharian-Sardroud A (2012) A linear-time algorithm for the longest path problem in rectangular grid graphs. Discrete Appl Math 160(3):210–217
14. Keshavarz-Kohjerdi F, Bagheri A (2012) Hamiltonian paths in some classes of grid graphs. J Appl Math. doi:10.1155/2012/475087
15. Lenhart W, Umans C (1997) Hamiltonian cycles in solid grid graphs. In: Proc 38th annual symposium on foundations of computer science (FOCS'97), pp 496–505
16. Loannidou K, Mertzios GB, Nikolopoulos S (2009) The longest path problem is polynomial on interval graphs. In: Proc of 34th int symp on mathematical foundations of computer science, Novy Smokovec, High Tatras, Slovakia, vol 5734. Springer, Berlin, pp 403–414
17. Luccio F, Mugnia C (1978) Hamiltonian paths on a rectangular chessboard. In: Proc 16th annual allerton conference, pp 161–173
18. Mertzios GB, Corneil DG A simple polynomial algorithm for the longest path problem on cocomparability graphs. SIAM J Discrete Math 26(3):940–963 (2012)
19. Myers BR (1981) Enumeration of tours in Hamiltonian rectangular latice graphs. Math Mag 54(1):19–23
20. Nandi M, Parui S, Adhikari A (2011) The domination numbers of cylindrical grid graphs. Appl Math Comput 217(10):4879–4889
21. Salman ANM (2005) Contributions to graph theory. PhD thesis, University of Twente
22. Uehara R, Uno Y (2007) On computing longest paths in small graph classes. Int J Found Comput Sci 18(5):911–930
23. Zamfirescu C, Zamfirescu T (1992) Hamiltonian properties of grid graphs. SIAM J Math 5(4):564–570
24. Zhang Z, Li H (2007) Algorithms for long paths in graphs. Theor Comput Sci 377(1–3):25–34
25. Zhang WQ, Liu YJ (2011) Approximating the longest paths in grid graphs. Theor Comput Sci 412(39):5340–5350