On Almost Monge All Scores Matrices*

Amir Carmel[1], Dekel Tsur[2], and Michal Ziv-Ukelson[3]

1 **Department of Computer Science, Ben-Gurion University of the Negev, Israel**
karmela@cs.bgu.ac.il
2 **Department of Computer Science, Ben-Gurion University of the Negev, Israel**
dekelts@cs.bgu.ac.il
3 **Department of Computer Science, Ben-Gurion University of the Negev, Israel**
michaluz@cs.bgu.ac.il

Abstract

The all scores matrix of a grid graph is a matrix containing the optimal scores of paths from every vertex on the first row of the graph to every vertex on the last row. This matrix is commonly used to solve diverse string comparison problems. All scores matrices have the Monge property, and this was exploited by previous works that used all scores matrices for solving various problems. In this paper, we study an extension of grid graphs that contain an additional set of edges, called bridges. Our main result is to show several properties of the all scores matrices of such graphs. We also give an $O(r(nm+n^2))$ time algorithm for constructing the all scores matrix of an $m \times n$ grid graph with r bridges.

1998 ACM Subject Classification F.2.0 Nonnumerical Algorithms and Problems

Keywords and phrases Sequence alignment, longest common subsequences, DIST matrices, Monge matrices, all path score computations.

Digital Object Identifier 10.4230/LIPIcs.CPM.2016.17

1 Introduction

String comparison is a fundamental problem in computer science that has applications in computational biology, computer vision, and other areas. String comparison is often performed using *sequence alignment*: The characters of two input strings are aligned to each other, and a *scoring function* gives a score to the alignment according to pairs of the aligned characters and unaligned characters. The goal of the string alignment problem is to seek an alignment that maximizes (or minimizes) the score. Common scoring functions are the *edit distance* score, and the *LCS* (longest common subsequence) score.

All scores matrices were introduced by Apostolico et al. [2] in order to obtain fast parallel algorithms for LCS computation. The *all scores matrix* of two strings A and B is a $(|B|+1) \times (|B|+1)$ matrix that stores the optimal alignment scores between A and every substring of B. More precisely, the element at row i and column j in the matrix is the optimal alignment score between A and $B[i..j]$. All scores matrices are also called DIST matrices [2] or semi-local score matrices [30].

The problem of efficiently constructing the all scores matrix of two strings has been studied in several papers [29, 1, 2, 17, 19, 20, 21, 22, 26, 31, 30]. All scores matrices provide a very powerful tool that can be also used for solving many problems on strings: optimal

* The research of A.C and D.T was partially supported by ISF grant 981/11. The research of A.C and M.Z-U was partially supported by ISF grant 478/10. The research of A.C, D.T, and M.Z-U was partially supported by the Frankel Center for Computer Science at Ben Gurion University of the Negev.

Figure 1 The crossing paths property yielding the Monge property in grid graphs. In Figure (a), the dark gray path is an optimal path from $(0,1)$ to $(4,5)$, and the light gray path is an optimal path from $(0,2)$ to $(4,4)$. These two paths cross at the vertex v. Figure (b) shows that a path from $(0,1)$ to $(4,4)$ can be obtained by taking the prefix of the dark gray path until v, and the suffix of the light gray path from v. Similarly, a path from $(0,2)$ to $(4,5)$ can be obtained by taking the prefix of the light gray path until v, and the suffix of the dark gray path from v. The sum of scores of the new paths is equal to the sum of scores of the former paths, which is equal to $D[1,5]+D[2,4]$. Since the new paths are not necessarily optimal, we obtain that $D[1,4]+D[2,5] \geq D[1,5]+D[2,4]$.

sequence alignment computation [9], approximate tandem repeats [24, 29], approximate non-overlapping repeats [5, 15, 29], common substring alignment [23, 25], sparse spliced alignment [16, 28], alignment of compressed strings [12], fully-incremental string comparison [14, 30], and other problems.

The alignment problem on strings A and B can be represented by using an $(|A|+1) \times (|B|+1)$ grid graph, known as the *alignment graph* (cf. [29]). Vertical (respectively, horizontal) edges correspond to alignment of a character in A (respectively, B) with a gap, and diagonal edges correspond to alignment of two characters in A and B. A path from the j-th vertex on row i to the j'-th vertex on row i' corresponds to an alignment of $A[i..i']$ and $B[j..j']$. The all scores matrix is therefore a matrix that contains the maximum (or minimum) scores of paths from vertices on the first row of the alignment graph to the vertices on the last row.

For an $n \times n$ matrix D, its *density* matrix $D^{\square}$ is an $(n-1) \times (n-1)$ matrix, where $D^{\square}[i,j] = D[i,j] + D[i-1,j-1] - D[i-1,j] - D[i,j-1]$. A matrix is called *Monge* if its density matrix is either non-negative or non-positive, and *unit Monge* if every row or column of the density matrix contains at most one non-zero element, and all the non-zero elements are equal to 1. All scores matrices of grid graphs are Monge matrices, this follows from the *crossing paths property* of the grid graph: If P_1 and P_2 are two paths from vertices on the first row to vertices on the last row of the graph, where on the first row the endpoint of P_1 appears before the endpoint of P_2, and on the last row the endpoint of P_1 appears after the endpoint of P_2, then the paths P_1 and P_2 must cross. This is illustrated in Figure 1. The Monge property is crucial for many of the algorithms for constructing all score matrices and for their applications. When the scoring function is the LCS score, the all scores matrix is unit Monge [31].

In this paper we extend the classical grid graphs to include an additional set of edges. These additional edges are of form $((i,j),(i',j'))$ where $i' \geq i$ and $j' \geq j$, and either $i' > i+1$ or $j' > j+1$ (see Figure 2a). We call these edges *bridges*. The bridges represent correspondence between pairs of substrings, one per each input sequence, which could be precomputed using an auxiliary adviser. In grid graphs enhanced with bridges, the crossing paths property no longer holds, and so the all scores matrix does not necessarily have the Monge property (see Figure 2).

Motivating examples of grid graphs enhanced with bridges are found in the domain of computational biology. Here, bridges are often used to incorporate additional information

that is known about the function and the physical structure of the aligned biomolecules and of their components [6, 11, 27]. One such example is found in a problem denoted "sequence alignment guided by motifs". Here, each one of the input sequences is first subjected to a parsing step in which meaningful substrings within it are identified and labeled. Substrings sharing the same label could be instantiations of the same motif shared by members of a protein family [13], particular DNA or RNA substrings of similar structure or function [4], or conserved molecular binding sites shared by multiple sequences that are combinatorially regulated in some biological pathway. Note that two substrings identified as belonging to the same motif family could be quite diverged in sequence, as it is the function, rather than the exact sequence, that is conserved in functional motifs. Yet, pairs of substrings sharing the same motif label are expected to be highly conserved in their location and order of occurrences within homologous genomic sequences. To incorporate this information, the alignment grid graph is enhanced with bridges reflecting pairs of substrings belonging to the same motif family, one from each sequence, and weights are assigned to these additional edges based on some a-priori scoring scheme expressing the importance of conserving the motifs in the alignment [4, 3, 8].

Our contribution and roadmap

In this paper, we consider grid graphs with bridges, and we assume that the non-bridge edges have 0/1 weights. We note that grid graphs with arbitrary bounded integer weights on the non-bridge edges can be reduced to grid graphs with 0/1 weights [30], and thus we will only consider the 0/1 weights scheme. However, this reduction is only quasi-polynomial: If the weights of non-bridge edges in the original grid graph are integers between $-C$ and C, the reduction increases the size of graph by a factor of $\Theta(C^2)$.

Our main result is to show the following properties of the non-zero values in the density matrix of an all scores matrix of a grid graph with r bridges (see Figure 2 for an example).

1. All the non-zero values in the density matrix are -1 or 1, except for $O(r^2)$ values in specific locations in the matrix.
2. In every row or column, except for r specific rows and r specific columns, the number of non-zero values is $O(r)$.

In particular, the number of non-zero values in the density matrix is $O(rn)$. Thus, if $r = o(n)$, the all scores matrix is "almost Monge". Property 1 will be proved in Section 2 (Theorem 3), and Property 2 in Sections 3. Due to space constraints, we only prove Property 2 for the case of a single bridge.

As a consequence of our main result, we obtain an algorithm for computing the all scores matrix for grid graphs with bridges in $O(r(nm + n^2))$ time. This algorithm is based on Schmidt's algorithm [29] for grid graphs with no bridges, and utilizes the properties described above. See below for comparison of this algorithm with previous results. The algorithm is given in Section 4 (Theorem 20).

Due to space constraints, some proofs were omitted.

Related work

Our algorithm mentioned above computes the optimal scores of paths from every vertex in a specific set of vertices (the vertices on the first row) to every vertex in the graph. This problem is called *multiple source shortest paths* (MSSP) problem. Algorithms for solving MSSP were proposed by several previous works. Schmidt [29] gave an MSSP algorithm for grid graphs with general weights. This algorithm constructs the all scores matrix in

$O((nm+n^2)\log n)$ time. For grid graphs with bounded integers weights, Schmidt gave an algorithm that constructs the all scores matrix in $O(mn)$ time. Tiskin [30] gave an MSSP algorithm for grid graphs with bounded integer weights that constructs the all scores matrix in $O(mn(\log\log n/\log n)^2)$ time. The results on grid graphs have been extended to general planar graphs. Klein [18] gave an algorithm for MSSP on planar graphs with general weights. The algorithm constructs the all scores matrix of a grid graph in $O((nm+n^2)\log n)$ time. Eisenstat and Klein [10] gave an algorithm for MSSP on undirected planar graphs with bounded integer weights, which is faster than the algorithm of Klein by a factor of $\Theta(\log n)$. Cabello et al [7] extended the result of Klein to graphs that can be embedded on a surface with genus g. Since a grid graph with r bridges can be embedded on a surface with genus r, the algorithm of Cabello et al. constructs the all scores matrix of a grid graph with r bridges and general weights in $O(rn^2\log^2 n)$ time. Cabello et al. also gave a randomized algorithm whose running time is $O(rn^2\log n)$ with high probability. Our algorithm improves the result of Cabello et al. by a factor of $\Theta(\log^2 n)$ for the case of bounded integer weights.

2 Preliminaries and basic problem properties

A *grid graph with bridges* is a directed graph $G=(V,E)$ whose vertex set is $V=\{(i,j): 0\le i\le m, 0\le j\le n\}$, and whose edge set consists of four types of edges:

1. Horizontal edges: $((i,j),(i,j+1))$ for every pair of indices i,j satisfying $0\le i\le m$ and $0\le j<n$.
2. Vertical edges: $((i,j),(i+1,j))$ for every pair of indices i,j satisfying $0\le i<m$ and $0\le j\le n$.
3. Diagonal edges: Edges of the form $((i,j),(i+1,j+1))$.
4. Bridges: Edges of the form $((i,j),(i',j'))$ where $i\le i'$ and $j\le j'$, and either $i+1<i'$ or $j+1<j'$.

In our framework, the horizontal and vertical edges have weight 0, the diagonal edges have weight 1, and each bridge has a positive integer weight. The *score* of a path is the sum of the weights of its edges. The 0/1 weights of the non-bridge edges correspond to the LCS scoring scheme for sequence alignment.

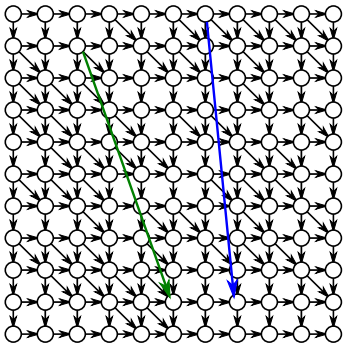
Let G be a grid graph with bridges $f_1,\ldots,f_r$. For a path P in G, if the first bridge P passes through is f_s, we say that P is an *s-path*. If P does not pass through bridges, we say that P is a *0-path*. The reason for focusing on the first bridge is to obtain a variant of the crossing path property which will be given in Lemma 12.

We define matrices D, $D^{\square}$, and D_{first} as follows (see Figure 2).

1. For $0\le i\le j\le n$, $D[i,j]$ is the maximum score of a path from $(0,i)$ to (m,j). For $i>j$, $D[i,j]=j-i$. The matrix D is called the *all scores matrix of G*.
2. For $1\le i,j\le n$, $D^{\square}[i,j]=(D[i,j]+D[i-1,j-1])-(D[i-1,j]+D[i,j-1])$. The matrix $D^{\square}$ is called the *density matrix of D*.
3. For $0\le i,j\le n$, $D_{\text{first}}[i,j]$ is a subset of the set $S=\{0,1,\ldots,r\}$ of bridge indices. For every $s\in S$, $s\in D_{\text{first}}[i,j]$ if and only if there is an s-path from $(0,i)$ to (m,j) with score $D[i,j]$.

To illustrate the importance of this matrix, consider a region in D_{first} in which all elements contain the same symbol s. Then, the crossing path property holds for indices in the region (since the two paths pass through f_s), so we obtain that the Monge property holds inside the region.

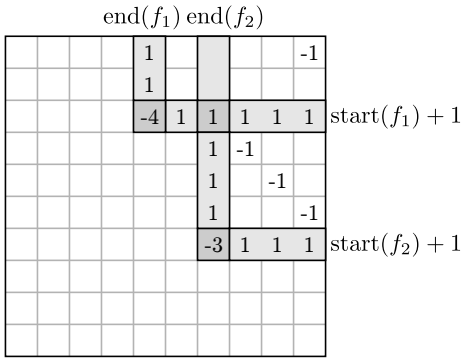
Next, we point out the entries in D and in $D^{\square}$ that are affected by a bridge in G. For some bridge $f_k=((i,j),(i',j'))$, we define $\text{start}(f_k)=j$ and $\text{end}(f_k)=j'$. We also define



(a) Grid graph

0	1	2	3	4	7	7	7	7	7	8
-1	0	1	2	3	7	7	7	7	7	7
-2	-1	0	1	2	7	7	7	7	7	7
-3	-2	-1	0	1	2	3	4	5	6	7
-4	-3	-2	-1	0	1	2	4	4	5	6
-5	-4	-3	-2	-1	0	1	4	4	4	5
-6	-5	-4	-3	-2	-1	0	4	4	4	4
-7	-6	-5	-4	-3	-2	-1	0	1	2	3
-8	-7	-6	-5	-4	-3	-2	-1	0	1	2
-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0

(b) D



(c) $D^{\square}$

0	0	0	0	0	1	1	1	1	0,1	0
	0	0	0	0	1	1	1	1	1	0,1
		0	0	0	1	1	1	1	1	0,1
			0	0	0	0	0,2	0	0	0
				0	0	0	2	0,2	0	0
					0	0	2	2	0,2	0
						0	2	2	2	0,2
							0	0	0	0
								0	0	0
									0	0
										0

(d) D_{first}

Figure 2 Figure (a) contains an example of a grid graph with two bridges. The weight of the bridge $f_1 = ((1,2),(9,5))$ is 7, and the weight of the bridge $f_2 = ((0,6),(9,7))$ is 4. The matrices D, $D^{\square}$, and D_{first} of the graph are shown in Figures (b), (c), and (d), respectively. Only the non-zero values of the density matrix are shown. The boundary indices are marked in grey, and the intersection indices are marked with darker grey. As stated in the text, each column or row of the density matrix can contain at most two negative values in non-boundary indices, and these values must be -1. The value of $D^{\square}[3,5]$ is $2+2-7-1=-4$, and these four values in D are marked in (b). The cause of the negative value in $D^{\square}[3,5]$ is that the corresponding optimal paths do not cross.

$E_k = \{(i,j) : i \leq \text{start}(f_k), j \geq \text{end}(f_k)\}$. In other words, E_k contains all indices (i,j) in D such that paths from $(0,i)$ to (m,j) can pass through f_k. The *boundary* of f_k is a set of indices in $D^{\square}$, defined as $B_k = \{(i, \text{end}(f_k)) : i \leq \text{start}(f_k)+1\} \cup \{(\text{start}(f_k)+1, j) : j \geq \text{end}(f_k)\}$. The two sets in the definition of B_k are called the *left boundary* and *bottom boundary* of f_k, respectively. We say that an index (i,j) in D is a *boundary index* if it is inside the boundary of some f_k. An index (i,j) is an *intersection index* if there are k, k' (possibly $k = k'$) such that (i,j) is in the left boundary of f_k and in the bottom boundary of $f_{k'}$ (see Figure 2c).

In the introduction we gave two properties of the density matrix. We now restate these properties using the definitions above.

1. Non-zero values other than $-1, 1$ can appear only at intersection indices.
2. In every row or column, the number of -1 values in non-boundary indices is at most r, and the number of 1 values in these indices is at most r.

Due to space constraints, we show these properties only for negative values. We will prove the properties for the positive values in the full version of the paper. Due to symmetry, we will show Property 2 only for columns.

Note that if $i > j+1$, $D^{\square}[i,j] = (j-i)+((j-1)-(i-1))-(j-(i-1))-((j-1)-i) = 0$. If $i = j+1$ then $D^{\square}[i,j] = -D[j,j]$, so in this case $D^{\square}[i,j] = 0$ unless there is a bridge f_k with $\text{start}(f_k) = \text{end}(f_k) = j$, in which case (i,j) is an intersection index. Similarly, for $i = j$, $D^{\square}[i,j] \in \{0,1\}$ unless one of the following two cases occurs: (1) There is a bridge f_k with $\text{start}(f_k) = j-1$ and $\text{end}(f_k) = j$. (2) There are bridges f_k and $f_{k'}$ with $\text{start}(f_k) = \text{end}(f_k) = j-1$ and $\text{start}(f_{k'}) = \text{end}(f_{k'}) = j$. In both cases (i,j) is an intersection index. Therefore, the properties stated above are satisfied for indices (i,j) with $i \geq j$. In the rest of the paper we will implicitly assume that indices (i,j) in $D^{\square}$ satisfy $i < j$.

We now give a proof for Property 1. For this goal, we need the following definition and lemma.

▶ **Definition 1.** A pair of indices $(i_1,j_1),(i_2,j_2)$ in the matrix D are said to be *bridge equivalent* if for every $1 \leq k \leq r$, $(i_1,j_1) \in E_k$ if and only if $(i_2,j_2) \in E_k$. In other words, $(i_1,j_1),(i_2,j_2)$ are bridge equivalent if paths from $(0,i_1)$ to (m,j_1) and paths from $(0,i_2)$ to (m,j_2) can pass through the same set of bridges.

▶ **Lemma 2.** *For every i,j,*
1. *If $(i,j-1)$ and (i,j) are bridge equivalent, $D[i,j-1] \leq D[i,j] \leq D[i,j-1]+1$.*
2. *If $(i-1,j)$ and (i,j) are bridge equivalent, $D[i,j] \leq D[i-1,j] \leq D[i,j]+1$.*

Property 1 is now obtained.

▶ **Theorem 3.** *Negative values other than -1 can appear only at intersection indices.*

Proof. Let (i,j) be an index that is not an intersection index. We have that either (1) $(i,j-1),(i,j)$ are bridge equivalent, and $(i-1,j-1),(i-1,j)$ are bridge equivalent, or (2) $(i-1,j),(i,j)$ are bridge equivalent, and $(i-1,j-1),(i,j-1)$ are bridge equivalent. In the former case we can rearrange the terms in the definition of $D^{\square}[i,j]$ and obtain that $D^{\square}[i,j] = \Delta_1 - \Delta_2$, where $\Delta_1 = D[i,j] - D[i,j-1]$ and $\Delta_2 = D[i-1,j] - D[i-1,j-1]$. We have $\Delta_1 - \Delta_2 < 0$, and by Lemma 2, $\Delta_1, \Delta_2 \in \{0,1\}$. It follows that $\Delta_1 = 0$ and $\Delta_2 = 1$, so $D^{\square}[i,j] = -1$. In the latter case we write $D^{\square}[i,j] = \Delta'_1 - \Delta'_2$ where $\Delta'_1 = D[i,j] - D[i-1,j]$ and $\Delta_2 = D[i,j-1] - D[i-1,j-1]$. By Lemma 2, in this case $\Delta'_1 = -1$ and $\Delta'_2 = 0$, so again $D^{\square}[i,j] = -1$. ◀

We next give several lemmas which will be used later to prove Property 2 in Section 3.

▶ **Definition 4.** An index (i,j) which is not a boundary index and for which $D^{\square}[i,j] < 0$ is called an *injury*. The submatrices $D[i-1..i, j-1..j]$ and $D_{\text{first}}[i-1..i, j-1..j]$ are called the submatrices of D and D_{first} corresponding to the injury, respectively.

▶ **Lemma 5.** *For an injury (i,j), $D[i-1..i, j-1..j] = \left(\begin{smallmatrix} x & x+1 \\ x & x \end{smallmatrix}\right)$ for some x.*

Proof. As in the proof of Theorem 3, $D^{\square}[i,j] = \Delta_1 - \Delta_2$, where $\Delta_1 = D[i,j] - D[i,j-1] = 0$ and $\Delta_2 = D[i-1,j] - D[i-1,j-1] = 1$. Thus, $D[i-1..i, j-1..j]$ is of the form $\left(\begin{smallmatrix} y & y+1 \\ x & x \end{smallmatrix}\right)$. We also have $D^{\square}[i,j] = \Delta'_1 - \Delta'_2$ where $\Delta'_1 = D[i,j] - D[i-1,j] = -1$ and $\Delta'_2 = D[i,j-1] - D[i-1,j-1] = 0$. The lemma follows. ◀

Our next goal is to show that every column in the density matrix contains at most r injuries. Consider a fixed column, and assume that this column has k injuries.

▶ **Definition 6.** Let $D_i = \left(\begin{smallmatrix} \gamma_i & \beta_i \\ \alpha_i & \delta_i \end{smallmatrix}\right)$ be the submatrix of D_{first} corresponding to the i-th injury, where the injuries are numbered in increasing row indices.

Our approach for proving that $k \leq r$ is based on showing properties of the D_{first} matrix. One of our techniques is showing that there are forbidden *structures* in D_{first}. For example, Lemma 10 below states that a structure consisting of a symbol $s \in \beta_i$ and $s \in \alpha_j$ for $j \geq i$ is forbidden. For the case of $r = 1$, applying this lemma with $i = j$ implies that there are only two possible values for α_i, β_i: either $\{0\}, \{1\}$ or $\{1\}, \{0\}$. If we assume conversely that there are $k = 2$ injuries, then there are four possible values for $\alpha_1, \beta_1, \alpha_2, \beta_2$. We then use Lemma 10 and an additional lemma (Lemma 12) that gives another forbidden structure in D_{first}, and show that each of these four cases cannot occur. This is a contradiction, and therefore there cannot be two injuries.

▶ **Lemma 7.** *For every i, j,*

1. *If $(i, j-1)$ and (i, j) are bridge equivalent,*
   - **(a)** *If $D[i, j-1] = D[i, j]$ then $D_{\text{first}}[i, j-1] \subseteq D_{\text{first}}[i, j]$.*
   - **(b)** *If $D[i, j-1] + 1 = D[i, j]$ then $D_{\text{first}}[i, j] \subseteq D_{\text{first}}[i, j-1]$.*
2. *If $(i-1, j)$ and (i, j) are bridge equivalent,*
   - **(a)** *If $D[i, j] = D[i-1, j]$ then $D_{\text{first}}[i, j] \subseteq D_{\text{first}}[i-1, j]$.*
   - **(b)** *If $D[i, j] + 1 = D[i-1, j]$ then $D_{\text{first}}[i-1, j] \subseteq D_{\text{first}}[i, j]$.*

Proof. We first prove the first part of the lemma. Choose a value $s \in D_{\text{first}}[i, j-1]$. Let P an s-path from $(0, i)$ to $(m, j-1)$ with score $D[i, j-1]$. The path P' obtained by appending the vertex (m, j) to P is an s-path from $(0, i)$ to (m, j) with score $D[i, j-1] = D[i, j]$ Therefore, $s \in D_{\text{first}}[i, j]$.

We next prove the second part of the lemma. Let $s \in D_{\text{first}}[i, j]$, and let P be an s-path from $(0, i)$ to (m, j) with score $D[i, j]$. Since $(i, j-1)$,(i, j) are bridge equivalent, P cannot pass through a bridge f with $\text{end}(f) > j-1$, so P has vertices on column $j-1$. Denote by k the maximal index such that $(k, j-1) \in P$. The path P' obtained by taking the prefix of P until $(k, j-1)$, and appending the vertices $(k+1, j-1), \ldots, (m, j-1)$ is an s-path from $(0, i)$ to $(m, j-1)$ with score at least $D[i, j] - 1 = D[i, j-1]$. It follows that $s \in D_{\text{first}}[i, j-1]$.

The proofs of the third and fourth parts of the lemma are symmetrical to the proofs of the first two parts, and thus they are omitted. ◀

The following lemma follows directly from Lemmas 5 and 7.

▶ **Lemma 8.** *For every i, $\alpha_i \subseteq \gamma_i \cap \delta_i$ and $\beta_i \subseteq \gamma_i \cap \delta_i$*

In order to restrict values of D in indices for which the entries in D_{first} contain the same symbol s, we define a matrix D_s as follows. For a symbol $s \in S$, let D_s be a matrix in which for every $(i, j) \in E_s$, $D_s[i, j]$ is the maximum score of an s-path from $(0, i)$ to (m, j). For $s = 0$, D_s is defined as above, except that $D_s[i, j]$ is defined for every $0 \leq i, j \leq n$. Note that $D_s[i, j] \leq D[i, j]$ for every (i, j) for which $D_s[i, j]$ is defined.

▶ **Lemma 9.** *For every $s \in S$, the matrix D_s has the Monge property.*

Proof. For $s = 0$ the lemma is true due to the crossing paths property for grid graphs with no bridges. For $s > 0$ we also have the crossing paths property: For every index (i, j), a maximum score s-path from $(0, i-1)$ to (m, j) must cross a maximum score s-path from $(0, i)$ to $(m, j-1)$ as both paths pass through f_s. Thus, the lemma follows. ◀

▶ **Lemma 10.** *For every $1 \leq i \leq j \leq k$, $\beta_i \cap \alpha_j = \emptyset$.*

Proof. Fix $i \leq j$, and assume conversely that $s \in \beta_i \cap \alpha_j$. By Lemma 5, the submatrices of D corresponding to injuries i and j are $D' = \left(\begin{smallmatrix} x & x+1 \\ x & \end{smallmatrix}\right)$ for some x, and $D'' = \left(\begin{smallmatrix} y & y+1 \\ y & \end{smallmatrix}\right)$ for some

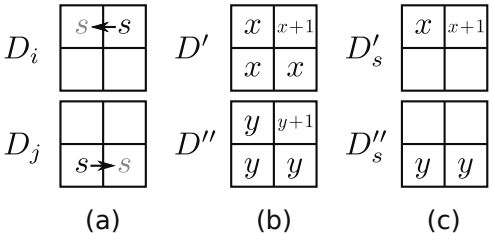


Figure 3 An illustration of the proof of Lemma 10. The grey s symbols in figure (a) represent values that are obtained using Lemma 8.

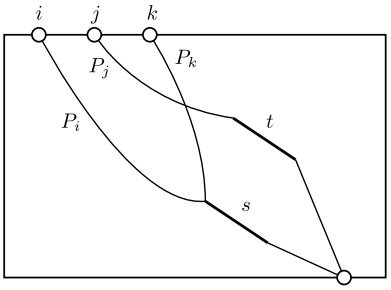


Figure 4 An illustration of the proof of Lemma 12.

y, respectively (see Figure 3). Let D'_s and D''_s be the submatrices of D_s that correspond to D' and D'', respectively. From the assumption $s \in \beta_i$ and Lemma 8, we have that $s \in \gamma_i$. Thus, the first row of D'_s is equal to the first row of D'. Similarly, we have that $s \in \delta_j$ and therefore the last row of D''_s is equal to the last row of D''. By taking the first row of D'_s and the last row of D''_s, we obtain that D_s contains a submatrix $\left(\begin{smallmatrix} x & x+1 \\ y & y \end{smallmatrix}\right)$ and therefore D_s does not have the Monge property. This contradicts Lemma 9. ◀

Finally, we give another forbidden structure in D_{first}, based on a variant of the crossing path property.

▶ **Definition 11.** Let $\preceq$ be a linear order on $S = \{0, 1, \ldots, r\}$ defined as follows. For every $i \neq j$, $i \preceq j$ if and only if $\text{start}(f_i) \leq \text{start}(f_j)$, where $\text{start}(f_0) = \infty$.

▶ **Lemma 12.** *Let d_i, d_j, d_k be values on rows i, j, k of some column i' of D_{first}, where $i < j < k$. Then, there are no $s, t \in S$ such that $s \preceq t$, $s \in d_i \cap d_k$, $t \notin d_i \cup d_k$, and $t \in d_j$.*

Proof. Assume conversely that there are $s, t \in S$ such that $s \preceq t$, $s \in d_i \cap d_k$, $t \notin d_i \cup d_k$, and $t \in d_j$. Note that $s \neq 0$ since by definition, $0 \not\preceq t$.

Let P_i, P_k be maximum score s-paths from $(0, i)$ and $(0, k)$ to (m, i'), respectively. Let P_j be a maximum score t-path from $(0, j)$ to (m, i'). Since $s \preceq t$, in the subgraph of G that contains the vertices above and to the left of the start vertex of f_s, the paths P_i, P_j, P_k do not pass through bridges (see Figure 4). Thus, P_j must cross one of the paths P_i and P_k. Assume without loss of generality that P_j crosses P_k.

Let P_j^1, P_k^1 denote the prefixes of P_j, P_k until the crossing point, and let P_j^2, P_k^2 denote the suffixes of P_j, P_k from the crossing point. Let y, z denote the scores of the paths P_j, P_k, respectively, and let a, b denote the score of the paths P_j^1, P_k^1, respectively.

We have that the path $P_k^1 \cup P_j^2$ is a t-path from $(0, k)$ to (m, i'). Since $t \notin d_k$, we conclude that $b + (y - a) < z$. Furthermore, due to the path $P_j^1 \cup P_k^2$ we have $a + (z - b) \leq y$. Summing the two inequalities above we obtain $y + z < y + z$, a contradiction. ◀

3 Properties of the one bridge case

In this section we assume the grid graph has a single bridge, $f = ((i_{\text{beg}}, j_{\text{beg}}), (i_{\text{end}}, j_{\text{end}}))$, and show that there is at most one injury in every column of $D^{\square}$.

▶ **Theorem 13.** *There is at most one injury in every column of $D^{\square}$.*

Proof. Fix some column of $D^{\square}$, and suppose conversely that there are at least two injuries in this column. Recall that $D_i = \left(\begin{smallmatrix} \gamma_i & \beta_i \\ \alpha_i & \delta_i \end{smallmatrix}\right)$ is the submatrix of D_{first} corresponding to the i-th

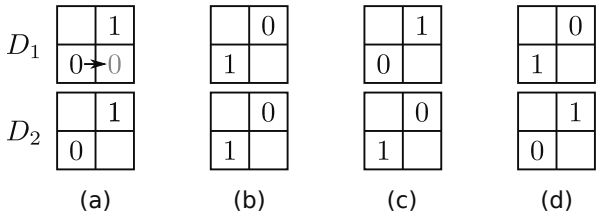


Figure 5 The four cases for two injuries in the proof of Theorem 13. The grey 0 in figure (a) represents a value that is obtained using Lemma 8.

injury. By Lemma 10, $\alpha_i \cap \beta_i = \emptyset$, and since α_i and β_i are non empty subsets of $S = \{0, 1\}$, it follows that either D_i is of the form $\left(\begin{smallmatrix} \cdot & 1 \\ 0 & \cdot \end{smallmatrix}\right)$ or D_i is of the form $\left(\begin{smallmatrix} \cdot & 0 \\ 1 & \cdot \end{smallmatrix}\right)$. Considering the first two injuries, there are four possible cases (see Figure 5):

1. D_1, D_2 are of the form $\left(\begin{smallmatrix} \cdot & 1 \\ 0 & \cdot \end{smallmatrix}\right)$.
2. D_1, D_2 are of the form $\left(\begin{smallmatrix} \cdot & 0 \\ 1 & \cdot \end{smallmatrix}\right)$.
3. D_1 is of the form $\left(\begin{smallmatrix} \cdot & 1 \\ 0 & \cdot \end{smallmatrix}\right)$ and D_2 is of the form $\left(\begin{smallmatrix} \cdot & 0 \\ 1 & \cdot \end{smallmatrix}\right)$.
4. D_1 is of the form $\left(\begin{smallmatrix} \cdot & 0 \\ 1 & \cdot \end{smallmatrix}\right)$ and D_2 is of the form $\left(\begin{smallmatrix} \cdot & 1 \\ 0 & \cdot \end{smallmatrix}\right)$.

We now show that each of the cases above yields a contradiction. In Case 1, we have from Lemma 8 that $0 \in \delta_1$. We now apply Lemma 12 on $\beta_1, \delta_1, \beta_2$ and obtain a contradiction (taking $s = 1$ and $t = 0$). Case 2 yields a contradiction using similar arguments. In Cases 3 and 4, we have $1 \in \beta_1 \cap \alpha_2$ and $0 \in \beta_1 \cap \alpha_2$, respectively, which is a contradiction to Lemma 10. ◀

Theorem 13 implies the following corollary.

▶ **Corollary 14.** *For $j \neq j_{\text{end}}$ there are at most two negative values in column j of $D^{\square}$. Moreover, the negative values can occur only in rows $1, \ldots, j_{\text{beg}} + 1$, and if there are two negative values, one of the values must be in row $j_{\text{beg}} + 1$.*

Proof. The column j can contain at most one injury. The column j has at most one boundary index, so there is at most one negative value in addition to the injury. ◀

4 Algorithm for constructing all-scores matrices

In this section we give an algorithm for computing the all scores matrix of a grid graph with bridges. Our algorithm is an extension of the algorithm of Schmidt for a grid graph without bridges [29]. We follow the presentation of Schmidt's algorithm which was given in Matarazzo et al. [26]. For clarity of presentation, we will first describe an algorithm for the case of a single bridge, and we will later handle the case of $r > 1$ bridges.

Let $f = ((i_{\text{beg}}, j_{\text{beg}}), (i_{\text{end}}, j_{\text{end}}))$ be the single bridge of the grid graph, and let W_f denote its weight.

Let $G_0, \ldots, G_m$ be grid graphs, where G_i is the subgraph of G induced by all the vertices (i', j) with $i' \leq i$. Let $D_0, \ldots, D_m$ be the all scores matrices of $G_0, \ldots, G_m$, respectively.

For $0 \leq k \leq n$, define

$$\textsc{DiffC}_{i,j}(k) = D_i[k, j+1] - D_i[k, j] \quad \text{and} \quad \textsc{DiffR}_{i,j}(k) = D_{i+1}[k, j] - D_i[k, j].$$

The following lemma follows from the definition above.

▶ **Lemma 15.** *For $i \leq m$, if all $\textsc{DiffC}_{i,j}(k)$ values are known for all j and k, then the matrix D_i can be constructed in $O(n^2)$ time.*

Our algorithm for constructing the all-scores matrix of G computes all $\textsc{DiffC}_{m,j}$ functions and then applies Lemma 15. The algorithm is based on the following properties of the $\textsc{DiffC}_{i,j}$ and $\textsc{DiffR}_{i,j}$ functions.

1. Most $\textsc{DiffC}_{i,j}$ and $\textsc{DiffR}_{i,j}$ functions have compact representations of size $O(1)$.
2. The compact representations of $\textsc{DiffC}_{i+1,j}$ and $\textsc{DiffR}_{i,j+1}$ can be computed efficiently from the compact representations of $\textsc{DiffC}_{i,j}$ and $\textsc{DiffR}_{i,j}$.

Property 1, stated in Lemma 18, is obtained from Lemmas 16 and 17 below. Property 2 is shown in Lemma 19. Similar properties were used in the algorithm of Schmidt for grid graphs with no bridges. In that case, *all* the $\textsc{DiffC}_{i,j}$ and $\textsc{DiffR}_{i,j}$ functions have compact representations, and the size of each representation is *exactly* 1. In the case of a grid graph with a bridge, we need additional steps to handle the $\textsc{DiffC}_{i,j}$ and $\textsc{DiffR}_{i,j}$ functions that do not have compact representations.

▶ **Lemma 16.** *For every $j \neq j_{\text{end}} - 1$, $\textsc{DiffC}_{i,j}(k) \in \{0, 1\}$, and for every $i \neq i_{\text{end}} - 1$, $\textsc{DiffR}_{i,j}(k) \in \{0, 1\}$.*

Proof. Follows immediately from Lemma 2. ◀

▶ **Lemma 17.**

1. *For every i and $j \neq j_{\text{end}} - 1$ there are $k_1 < k_2$ (where $k_2 = j_{\text{beg}} + 1$) such that for every $k \neq k_1, k_2$, $\textsc{DiffC}_{i,j}(k-1) \leq \textsc{DiffC}_{i,j}(k)$.*
2. *For every $i \neq i_{\text{end}} - 1$ and j there are $k_1 < k_2$ (where $k_2 = j_{\text{beg}} + 1$) such that for every $k \neq k_1, k_2$, $\textsc{DiffR}_{i,j}(k-1) \geq \textsc{DiffR}_{i,j}(k)$.*

Based on the previous two lemmas, we now give a compact representation for the $\textsc{DiffR}_{i,j}$ and $\textsc{DiffC}_{i,j}$ functions. The compact representation $\text{SR}_{i,j}$ of $\textsc{DiffR}_{i,j}$ is an array of "step" indices, i.e., the indices in which the value of $\textsc{DiffR}_{i,j}$ change. Formally, let I be the set of all indices k such that $\textsc{DiffR}_{i,j}(k) \neq \textsc{DiffR}_{i,j}(k-1)$. Then, $\text{SR}_{i,j}[l]$ is the l-th smallest element of I. The arrays $\text{SC}_{i,j}$ are defined similarly.

▶ **Lemma 18.** *For every $i \neq i_{\text{end}} - 1$ and $j \neq j_{\text{end}} - 1$, the arrays $\text{SR}_{i,j}$ and $\text{SC}_{i,j}$ have $O(1)$ elements each.*

In the following lemma we show that $\text{SC}_{i+1,j}$ and $\text{SR}_{i,j+1}$ can be computed efficiently from $\text{SC}_{i,j}$ and $\text{SR}_{i,j}$. For every $(i,j) \neq (i_{\text{end}} - 1, j_{\text{end}} - 1)$ and $k \leq j$, the optimal path from $(0,k)$ to $(i+1, j+1)$ passes through either $(i+1, j)$, (i,j), or $(i, j+1)$. Thus,

$$D_{i+1}[k, j+1] = \max\{D_{i+1}[k,j], D_i[k,j] + W_{i,j}, D_i[k, j+1]\},$$

where $W_{i,j} = 1$ if there is a diagonal edge entering (i,j) and $W_{i,j} = 0$ otherwise. From the equality above, the following formulas for $\textsc{DiffC}_{i+1,j}$ and $\textsc{DiffR}_{i,j+1}$ are obtained (see [26]).

▶ **Lemma 19.** *For $0 \leq k \leq j$ and $(i,j) \neq (i_{\text{end}} - 1, j_{\text{end}} - 1)$,*

$$\textsc{DiffC}_{i+1,j}(k) = \textsc{Max}_{i,j}(k) - \textsc{DiffR}_{i,j}(k) \text{ and } \textsc{DiffR}_{i,j+1}(k) = \textsc{Max}_{i,j}(k) - \textsc{DiffC}_{i,j}(k)$$

where $\textsc{Max}_{i,j}(k) = \max\{\textsc{DiffR}_{i,j}(k), W_{i,j}, \textsc{DiffC}_{i,j}(k)\}$.

We will use compact representations $\text{SMAX}_{i,j}$ for the $\textsc{Max}_{i,j}$ functions, which are defined similarly to the $\text{SR}_{i,j}$ arrays. From the definition of $\textsc{Max}_{i,j}$, every step of $\textsc{Max}_{i,j}$ corresponds to a step of either $\textsc{DiffC}_{i,j}$ or $\textsc{DiffR}_{i,j}$, and thus the number of elements in $\text{SMAX}_{i,j}$ is less then or equal to the number of elements in both $\text{SC}_{i,j}$ and $\text{SR}_{i,j}$. Therefore, $\text{SMAX}_{i,j}$ has $O(1)$ elements for $i \neq i_{\text{end}} - 1$ and $j \neq j_{\text{end}} - 1$.

Our algorithm for computing the arrays $\mathrm{SC}_{m,j}$, traverses every i, j and computes $\mathrm{SC}_{i+1,j}$ and $\mathrm{SR}_{i,j+1}$ from $\mathrm{SC}_{i,j}$ and $\mathrm{SR}_{i,j}$ using Lemma 19. When $i \neq i_{\mathrm{end}} - 1$ and $j \neq j_{\mathrm{end}} - 1$, this computation takes $O(1)$ time by Lemma 18. There are two cases which require a special treatment. The first case is $(i, j) = (i_{\mathrm{end}} - 1, j_{\mathrm{end}} - 1)$. In this case Lemma 19 can not be applied and thus $\mathrm{SC}_{i+1,j}$ and $\mathrm{SR}_{i,j+1}$ must be computed differently. Here we compute $D_{i+1}[k, j]$, $D_i[k, j+1]$, and $D_{i+1}[k, j+1]$, for every $0 \leq k \leq n$. Then, we use these values to compute $\textsc{DiffC}_{i+1,j}(k)$ and $\textsc{DiffR}_{i,j+1}(k)$ for all k, and finally we compute $\mathrm{SC}_{i+1,j}$ and $\mathrm{SR}_{i,j+1}$ from $\textsc{DiffC}_{i+1,j}$ and $\textsc{DiffR}_{i,j+1}$.

The values $D_{i+1}[k, j]$ and $D_i[k, j+1]$ are obtained using Lemma 15 in $O(n^2)$ time. To compute the $D_{i+1}[k, j+1]$ values, we use the equality

$$D_{i+1}[k, j+1] = \max\{D_i[k, j+1], D_i[k, j] + W_{i,j}, D_{i+1}[k, j], D_{i_{\mathrm{beg}}}[k, j_{\mathrm{beg}}] + W_f\}.$$

The second special case is when $i = i_{\mathrm{end}} - 1$ or $j = j_{\mathrm{end}} - 1$. In this case Lemma 18 does not apply. Therefore, we can only bound the time to compute $\mathrm{SC}_{i+1,j}$ and $\mathrm{SR}_{i,j+1}$ by $O(n)$. Since there are $O(n + m)$ pairs i, j for which this case occurs, the total contribution of this case to the time complexity of the algorithm is $O(n^2 + nm)$.

Extension to r bridges

The algorithm presented above can be extended to the case of $r > 1$ bridges. In this case, using the results of the next section we get that for every non-boundary pair (i, j), $\textsc{DiffC}_{i,j}$ and $\textsc{DiffR}_{i,j}$ are partitioned to $O(r)$ monotone regions and thus their compact representations $\mathrm{SC}_{i,j}, \mathrm{SR}_{i,j}$ have $O(r)$ elements. Therefore, the computation of $\mathrm{SC}_{i,j}, \mathrm{SR}_{i,j}$ for non-boundary indices takes $O(rnm)$ time. As for boundary indices, the technique remains as in the case of one bridge, only that now there are $O(r)$ intersection indices and $O(r(n+m))$ boundary indices. Summing the above, the following theorem is obtained.

▶ **Theorem 20.** *The all scores matrix for an $m \times n$ grid graph with r bridges can be constructed in $O(r(nm + n^2))$ time.*

Acknowledgments. We thank the anonymous CPM 2016 reviewers for their helpful comments.

References

1. C. E. R. Alves, E. N. Cáceres, and S. W. Song. An all-substrings common subsequence algorithm. *Discrete Applied Mathematics*, 156(7):1025–1035, 2008.
2. A. Apostolico, M. J. Atallah, L. L. Larmore, and S. McFaddin. Efficient parallel algorithms for string editing and related problems. *SIAM J. on Computing*, 19(5):968–988, 1990.
3. A. N. Arslan. Sequence alignment guided by common motifs. In *Proceedings of the fourth Biotechnology and Bioinformatics Symposium*, page 30. University of Colorado at Colorado Springs, 2007.
4. A. N. Arslan. Sequence alignment guided by common motifs described by context free grammars. In *Biotechnology and Bioinformatics Symp. (BIOT'07)*, 2007.
5. G. Benson. A space efficient algorithm for finding the best nonoverlapping alignment score. *Theoretical Computer Science*, 145(1&2):357–369, 1995.
6. H. L. Bodlaender, M. R. Fellows, and P. A. Evans. Finite-state computability of annotations of strings and trees. In *Combinatorial Pattern Matching*, pages 384–391. Springer, 1996.
7. S. Cabello, E. W. Chambers, and J. Erickson. Multiple-source shortest paths in embedded graphs. *SIAM J. on Computing*, 42(4):1542–1571, 2013.
8. J.-P. Comet and J. Henry. Pairwise sequence alignment using a prosite pattern-derived similarity score. *Computers & chemistry*, 26(5):421–436, 2002.

9 M. Crochemore, G. M. Landau, and M. Ziv-Ukelson. A sub-quadratic sequence alignment algorithm for unrestricted cost matrices. *SIAM J. on Computing*, 32(5):1654–1673, 2003.

10 D. Eisenstat and P. N. Klein. Linear-time algorithms for max flow and multiple-source shortest paths in unit-weight planar graphs. In *Proc. 45th ACM Symposium on Theory Of Computing (STOC)*, pages 735–744, 2013.

11 P. A. Evans. *Algorithms and complexity for annotated sequence analysis*. PhD thesis, Citeseer, 1999.

12 D. Hermelin, G. M. Landau, S. Landau, and O. Weimann. A unified algorithm for accelerating edit-distance computation via text-compression. In *Proc. 26th Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 529–540, 2009.

13 N. Hulo, A. Bairoch, V. Bulliard, L. Cerutti, E. De Castro, P. S. Langendijk-Genevaux, M. Pagni, and C. Sigrist. The prosite database. *Nucleic acids research*, 34(suppl 1):D227–D230, 2006.

14 Y. Ishida, S. Inenaga, A. Shinohara, and M. Takeda. Fully incremental LCS computation. In *Proc. 15th Symp. on Fundamentals of Computation Theory (FCT)*, pages 563–574, 2005.

15 S. Kannan and E. W. Myers. An algorithm for locating nonoverlapping regions of maximum alignment score. *SIAM J. on Computing*, 25(3):648–662, 1996.

16 C. Kent, G. M. Landau, and M. Ziv-Ukelson. On the complexity of sparse exon assembly. *Journal of Computational Biology*, 13(5):1013–1027, 2006.

17 S.-R. Kim and K. Park. A dynamic edit distance table. *J. of Discrete Algorithms*, 2(2):303–312, 2004.

18 P. N. Klein. Multiple-source shortest paths in planar graphs. In *Proc. 16th Symposium on Discrete Algorithms (SODA)*, volume 5, pages 146–155, 2005.

19 P. Krusche and A. Tiskin. String comparison by transposition networks. In *Proc. of London Algorithmics Workshop*, pages 184–204, 2008.

20 P. Krusche and A. Tiskin. New algorithms for efficient parallel string comparison. In *Proc. 22nd Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 209–216, 2010.

21 G. M. Landau, E. W. Myers, and J. P. Schmidt. Incremental string comparison. *SIAM J. on Computing*, 27(2):557–582, 1998.

22 G. M. Landau, E. W. Myers, and M. Ziv-Ukelson. Two algorithms for LCS consecutive suffix alignment. *J. of Computer and System Sciences*, 73(7):1095–1117, 2007.

23 G. M. Landau, B. Schieber, and M. Ziv-Ukelson. Sparse LCS common substring alignment. *Information Processing Letters*, 88(6):259–270, 2003.

24 G. M. Landau, J. P. Schmidt, and D. Sokol. An algorithm for approximate tandem repeats. *J. of Computational Biology*, 8(1):1–18, 2001.

25 G. M. Landau and M. Ziv-Ukelson. On the common substring alignment problem. *J. of Algorithms*, 41(2):338–359, 2001.

26 U. Matarazzo, D. Tsur, and M. Ziv-Ukelson. Efficient all path score computations on grid graphs. *Theoretical Computer Science*, 525:138–149, 2014.

27 S. Mozes, D. Tsur, O. Weimann, and M. Ziv-Ukelson. Fast algorithms for computing tree lcs. *Theoretical Computer Science*, 410(43):4303–4314, 2009.

28 Y. Sakai. An almost quadratic time algorithm for sparse spliced alignment. *Theory of Computing Systems*, pages 1–22, 2009.

29 J. P. Schmidt. All highest scoring paths in weighted grid graphs and their application to finding all approximate repeats in strings. *SIAM J. of Computing*, 27(4):972–992, 1998.

30 A. Tiskin. Semi-local string comparison: algorithmic techniques and applications. arXiv:0707.3619v16.

31 A. Tiskin. Semi-local longest common subsequences in subquadratic time. *J. Discrete Algorithms*, 6(4):570–581, 2008.