Theoretical Computer Science 525 (2014) 138–149

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Efficient all path score computations on grid graphs

Ury Matarazzo[1], Dekel Tsur[1], Michal Ziv-Ukelson[*,2]

Department of Computer Science, Ben-Gurion University of the Negev, Israel

ARTICLE INFO

Keywords:
Sequence alignment
All path score computations

ABSTRACT

We study the *Integer-weighted Grid All Paths Scores* (IGAPS) problem, which is given a grid graph, to compute the maximum weights of paths between every pair of a vertex on the first row of the graph and a vertex on the last row of the graph. We also consider a variant of this problem, periodic IGAPS, where the input grid graph is periodic and infinite. For these problems, we consider both the general (dense) and the sparse cases.
For the sparse periodic IGAPS problem with 0–1 weights, we give an $O(r\log^3(n^2/r))$ time algorithm, where r is the number of (diagonal) edges of weight 1. Our result improves upon the previous $O(n\sqrt{r})$ result by Krusche and Tiskin for this problem.
For the periodic IGAPS problem we give an $O(Cn^2)$ time algorithm, where C is the maximum weight of an edge. This improves upon the previous $O(C^2n^2)$ algorithm of Tiskin. We also show a reduction from periodic IGAPS to IGAPS. This reduction yields $o(n^2)$ algorithms for this problem.

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 *string 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. In this paper we consider maximal scores to be optimal, but minimization problems can be solved symmetrically. The problem can be solved in $O(n^2)$ time [27], where n is the sum of lengths of A and B. Common scoring functions are the *edit distance* score, and the *LCS* (longest common subsequence) score.

A *grid graph* (see Fig. 1) is a directed graph $G=(V,E)$ whose vertex set is $V=\{(i,j)\colon\ 0\leqslant i\leqslant m,\ 0\leqslant j\leqslant n\}$, and whose edge set consists of three types:

1. Diagonal edges $((i,j),(i+1,j+1))$ for all $0\leqslant i<m, 0\leqslant j<n$.
2. Horizontal edges $((i,j),(i,j+1))$ for all $0\leqslant i\leqslant m, 0\leqslant j<n$.
3. Vertical edges $((i,j),(i+1,j))$ for all $0\leqslant i<m, 0\leqslant j\leqslant n$.

An example of a grid graph can be found in Fig. 1. In the *Grid All Paths Scores* (GAPS) problem, the input is a grid graph and the goal is to compute the maximum weights of paths between every pair of a vertex on the first row of the graph and a vertex on the last row of the graph. For simplicity of presentation, we will assume in some parts of this paper that $m=n$.

[*] Corresponding author.
E-mail addresses: ury@cs.bgu.ac.il (U. Matarazzo), dekelts@cs.bgu.ac.il (D. Tsur), michaluz@cs.bgu.ac.il (M. Ziv-Ukelson).
[1] The research of D.T. and U.M. was partially supported by ISF grant 981/11.
[2] The research of M.Z.U. was partially supported by ISF grant 478/10.

0304-3975/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.tcs.2013.07.018

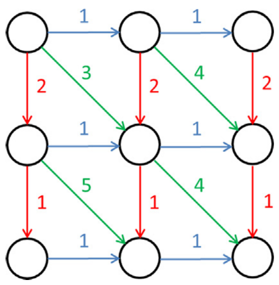

Fig. 1. A grid graph of size 2×2.

The *Integer GAPS* (IGAPS) problem is a special case of GAPS in which the weights of the edges are integers in the range 0 to C, and additionally, the weights of all the horizontal (resp., vertical) edges between two columns (resp., rows) of vertices are equal. The *Binary GAPS* (BGAPS) problem is a special case of IGAPS in which the horizontal and vertical edges have weight 0, and diagonal edges have weight 0 or 1.

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 grid graph* (cf. [21]). 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. Each edge of the graph has a weight. 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 GAPS problem was introduced by Apostolico et al. [3] in order to obtain fast parallel algorithms for LCS computation. It has since been studied in several additional papers [1,2,7,11–15,21–23]. Schmidt [21] showed that the GAPS problem can be solved in $O(n^2 \log n)$ time. In the same paper, Schmidt showed that IGAPS can be solved in $O(Cn^2)$ time. An $O(n^2)$ algorithm based on a similar approach for the BGAPS problem was also given by Alves et al. [1] and Tiskin [23]. Tiskin [22, p. 60] gave an $O(n^2(\log\log n/\log n)^2)$ time algorithm for a special case of BGAPS, in which the grid graph corresponds to an LCS problem on two strings. Tiskin also showed that IGAPS can be reduced to BGAPS. However, this reduction increases the size of the grid graph by a factor of C^2. Thus, the time for solving IGAPS with this reduction is either $O(C^2n^2)$ (for general grid graphs) or $O(C^2n^2(\log\log n/\log n)^2)$ (for grid graphs that correspond to alignment problems on two strings).

A special case of the BGAPS problem is when the number of diagonal edges with weight 1 is significantly smaller than n^2. We call this problem *sparse BGAPS*. Krusche and Tiskin [12] showed that sparse BGAPS can be solved in $O(n\sqrt{r})$ time, where r is the number of edges of weight 1. For the special case of a permutation grid graph (namely, each column and each row have exactly one edge of weight 1), Tiskin [22] gave an $O(n\log^2 n)$ time algorithm. Another special case of BGAPS is when the grid graph corresponds to the LCS computation of two strings with little similarity. Landau et al. [15] gave an algorithm for this variant with time complexity $O(nL)$, where L is the LCS of the two strings.

Efficient computations and storage of GAPS provide very powerful tools that can be also used for solving many problems on strings: optimal alignment computation [5], approximate tandem repeats [17,21], approximate non-overlapping repeats [4,9,21], common substring alignment [16,18], sparse spliced alignment [10,20], alignment of compressed strings [6], fully-incremental string comparison [8,19,22], and other problems.

Additional types of computations are useful in some of the applications. A *periodic grid graph* is an infinite graph obtained by concatenating horizontally an infinite number of a (finite) grid graph. The *periodic IGAPS* problem is a variant of the IGAPS problem, in which the input is a periodic grid graph. Note that while there are an infinite number of vertex pairs whose maximum path score need to be computed, due to the periodicity of the graph, the output can be represented in finite space. The periodic IGAPS problem was studied by Tiskin [25] who gave an $O(C^2n^2)$ time algorithm for the problem.

1.1. Our contribution and road map

In this work we address several variants of the IGAPS problem. Our contribution includes generalizations and improvements to previous results as follows (summarized in Table 1).

We start by working out some of the previously vague details from Schmidt's algorithm [21] for a special case of the IGAPS problem (the assumption in [21] is that all horizontal and vertical edges have weight w_1, and each diagonal edge has weight w_1 or w_2, for some fixed w_1 and w_2). We generalize Schmidt's algorithm to yield an $O(Cn^2)$ algorithm for the general IGAPS problem (Section 3).

In Section 4 we consider the sparse BGAPS problem. We give an $O(r\log^3(n^2/r))$ time algorithm, which improves the previous result of Krusche and Tiskin for this problem.

Next, we turn to address the periodic IGAPS problem in Section 5. Our first result on this front is obtained by extending the $O(Cn^2)$ algorithm for IGAPS to handle the periodic variant of the problem (Section 5.1). This improves Tiskin's $O(C^2n^2)$ result for periodic IGAPS. We then show, in Section 5.2, that periodic IGAPS can be reduced to BGAPS. Therefore, we obtain an $O(C^2n^2(\log\log n/\log n)^2)$ time algorithm for periodic IGAPS (when the grid graph corresponds to an alignment problem), and an $O(r\log^3(n^2/r))$ time algorithm for periodic sparse BGAPS.