Abstract

Let T be a terrain, and let P be a set of points (locations) on its surface. An important problem in Geographic Information Science (GIS) is computing the *visibility index* of a point p on P, that is, the number of points in P that are visible from p. The *total visibility-index* problem asks for computing the visibility index of *every* point in P. Most applications of this problem involve 2-dimensional terrains represented by a grid of $n \times n$ square cells, where each cell is associated with an elevation value, and P consists of the center-points of these cells. Current approaches for computing the total visibility-index on such a terrain take at least quadratic time with respect to the number of the terrain cells. While finding a subquadratic solution to this *2D total visibility-index* problem is an open problem, surprisingly, no subquadratic solution has been proposed for the one-dimensional (1D) version of the problem; in the 1D problem, the terrain is an x-monotone polyline, and P is the set of the polyline vertices. We present an $O(n \log^2 n)$ algorithm that solves the 1D total visibility-index problem in the RAM model. Our algorithm is based on a geometric dualization technique, which reduces the problem into a set of instances of the red-blue line segment intersection counting problem. We also present a parallel version of this algorithm, which requires $O(\log^2 n)$ time and $O(n \log^2 n)$ work in the CREW PRAM model. We implement a naive $O(n^2)$ approach and three variations of our algorithm: one employing an existing red-blue line segment intersection algorithm and two new approaches that perform the intersection counting by leveraging features specific to our problem. We present experimental results for both serial and parallel implementations on large synthetic and real-world datasets, using two distinct hardware platforms. Results show that all variants of our algorithm outperform the naive approach by several orders of magnitude on large datasets. Furthermore, we show that our new intersection counting implementations achieve more than 8 times speedup over the existing red-blue line segment intersection algorithm. Our parallel implementation is able to process a terrain of 2^{24} vertices in under 1 minute using 16 cores, achieving more than 7 times speedup over serial execution.