Preprocess, Set, *Query!*

Ely Porat and Liam Roditty*

Department of Computer Science, Bar-Ilan University, Ramat-Gan, Israel
{poratelg,liamr}@macs.biu.ac.il

Abstract. Thorup and Zwick [J. ACM and STOC'01] in their seminal work introduced the notion of *distance oracles*. Given an n-vertex weighted undirected graph with m edges, they show that for any *integer* $k \geq 1$ it is possible to preprocess the graph in $\tilde{O}(mn^{1/k})$ time and generate a compact data structure of size $O(kn^{1+1/k})$. For each pair of vertices, it is then possible to retrieve an estimated distance with multiplicative stretch $2k-1$ in $O(k)$ time. For $k=2$ this gives an oracle of $O(n^{1.5})$ size that produces in constant time estimated distances with stretch 3. Recently, Pătraşcu and Roditty [FOCS'10] broke the long-standing theoretical status-quo in the field of distance oracles and obtained a distance oracle for sparse unweighted graphs of $O(n^{5/3})$ size that produces in constant time estimated distances with stretch 2.

In this paper we show that it is possible to break the stretch 2 barrier at the price of non-constant query time. We present a data structure that produces estimated distances with $1+\varepsilon$ stretch. The size of the data structure is $O(nm^{1-\varepsilon'})$ and the query time is $\tilde{O}(m^{1-\varepsilon'})$. Using it for sparse unweighted graphs we can get a data structure of size $O(n^{1.86})$ that can supply in $O(n^{0.86})$ time estimated distances with multiplicative stretch 1.75.

1 Introduction

Thorup and Zwick [15] initiated the theoretical study of data structures capable of representing almost shortest paths efficiently, both in terms of space requirement and query time. Given an n-vertex weighted undirected graph with m edges, they show that for any integer $k \geq 1$ it is possible to preprocess the graph in $\tilde{O}(mn^{1/k})$ time and generate a compact data structure of size $O(kn^{1+1/k})$. For each pair of vertices, it is then possible to retrieve an estimated distance with multiplicative stretch $2k-1$ in $O(k)$ time. An estimated distance has multiplicative (additive) stretch c if for two vertices at distance Δ it is at least Δ and at most $c\Delta$ $(\Delta+c)$. We use (α, β) to denote a combination of multiplicative stretch of α and additive stretch of β, that is, the estimated distance is at most $\alpha\Delta+\beta$. Thorup and Zwick called such data structures *distance oracles* as their query time is constant for any constant stretch. Thorup and Zwick [15] showed also that their data structure is optimal for dense graphs. Based on the girth conjecture of Erdős they showed that there exist dense enough graphs which cannot

* Work supported by Israel Science Foundation (grant no. 822/10).

C. Demetrescu and M.M. Halldórsson (Eds.): ESA 2011, LNCS 6942, pp. 603–614, 2011.
© Springer-Verlag Berlin Heidelberg 2011

be represented by a data structure of size less than $n^{1+1/k}$ without increasing the stretch above $2k-1$ for any integral k.

Sommer, Verbin, and Yu [13] proved a three-way tradeoff between space, stretch and query time of approximate distance oracles. They show that any distance oracle that can give stretch α answers to distance queries in time $O(t)$ must use $n^{1+\Omega(1/(t\alpha))}/\log n$ space. Their result is obtained by a reduction from lopsided set disjointness to distance oracles, using the framework introduced by Pătraşcu [9]. Any improvement to this lower bound requires a major breakthrough in lower bounds techniques. In particular, it does not imply anything even for slightly non-constant query time as $\Omega(\log n)$ and slightly non-linear space as $n^{1.01}$.

This suggests that upper bounds are currently the only realistic way to attack the Thorup and Zwick space-stretch-query tradeoff. There are several possible ways to obtain a progress:

1. To consider sparse graphs. In particular, graphs with less than $n^{1+1/k}$ edges.
2. To consider also additive error, that is, to get below $2k-1$ multiplicative stretch with an additional additive stretch.
3. Non-constant query time.

The first way and the second one are closely related. We cannot gain from introducing also additive stretch without getting an improved multiplicative stretch for sparse graphs (i.e., $m = O(n)$). A data structure with size $S(m,n)$ and stretch (α,β) implies a data structure with size $S((\beta+1)m, n+\beta m)$ and multiplicative stretch of α, as if we divide every edge into $\beta+1$ edges then all distances become a multiply of $\beta+1$ and additive stretch of β is useless. For graphs with $m = O(n)$ the size of the data structure is asymptotically the same.

Recently, Pătraşcu and Roditty [10] broke the long-standing theoretical status-quo in the field of distance oracles. They obtained a distance oracle for sparse unweighted graphs of size $O(n^{5/3})$ that can supply in $O(1)$ time an estimated distance with multiplicative stretch 2. For dense graphs the distance oracle has the same size and stretch $(2,1)$. Pătraşcu and Roditty [10] showed also a conditional lower bound for distance oracle that is based on a conjecture on the hardness of the set intersection problem. They showed that a distance oracle for unweighted graphs with $m = \tilde{O}(n)$ edges, which can distinguish between distances of 2 and 4 in constant time (as multiplicative stretch strictly less than 2 implies) requires $\tilde{\Omega}(n^2)$ space, assuming the conjecture holds. Thus, non-constant query time is essential to get stretch smaller than 2.

In this paper we show that one can gain by allowing non-constant query time. We break the barrier of 2 for multiplicative stretch at the price of non-constant query time. Surprisingly, we show that it is possible to get an arbitrary small multiplicative stretch.

We prove the following:

Theorem 1. *For any unweighted graph and any $\varepsilon > 0$, we can construct in $O(mn)$ time a data structure of size $O(nm^{1-\frac{\varepsilon}{4+2\varepsilon}})$ that given any two vertices s and t returns an estimated distance with multiplicative stretch $1+\varepsilon$ in*