On Computing the Diameter of Real-World Directed (Weighted) Graphs

Pierluigi Crescenzi[1], Roberto Grossi[2], Leonardo Lanzi[1], and Andrea Marino[1]

[1] Dipartimento di Sistemi e Informatica, Università degli Studi di Firenze, Italy
[2] Dipartimento di Informatica, Università degli Studi di Pisa, Italy

Abstract. In this paper we propose a new algorithm for computing the diameter of directed unweighted graphs. Even though, in the worst case, this algorithm has complexity $O(nm)$, where n is the number of nodes and m is the number of edges of the graph, we experimentally show that in practice our method works in $O(m)$ time. Moreover, we show how to extend our algorithm to the case of directed weighted graphs and, even in this case, we present some preliminary very positive experimental results.

1 Introduction

The analysis of real-world networks such as biological, collaboration, communication, road, social, and web networks has attracted a lot of attention in the last two decades, and many properties of these networks have been studied (see, for example, [21,5,12]). Since the size of real-world networks has been increasing rapidly, in order to study these properties, we need algorithms that can handle huge amount of data. In this paper we will focus our attention on a very basic property of real-world networks, that is, their diameter. Given a directed graph $G = (V, E)$, the diameter of G is the minimum D such that, for any pair of nodes $u, v \in V$, the *distance* $d(u, v)$ between them is at most D, where $d(u, v)$ is the length of the shortest path from u to v (whenever the graph includes a pair of nodes u, v such that $d(u, v) = \infty$, we will study the diameter of its largest strongly connected component).

The diameter is a relevant measure whose meaning depends on the semantics of the real-world network. In the case of social networks, in which every node is an individual and the edges represent their social relationships, the diameter can indicate how quickly information reaches every individual in the worst case, and it has been studied for several social networks (see, for example, [26,20,18]). In the case of web networks, in which every node corresponds to a web page and the edges correspond to hyper-links, the diameter indicates how quickly (in terms of mouse clicks) any page can be reached in the worst case: for several web networks, the diameter has been considered, for example, in [6,18,15]. In the case of communication networks, in which every node is a device and the edges represent communication links, the diameter indicates, for example, the completion time of broadcast protocols based on network flooding. In the case of biological networks, finally, the cellular metabolism is represented by a network of metabolites linked by biochemical reactions, and the diameter indicates how many reactions have to be performed, in the worst case, in order to produce any metabolite from any other metabolite [13]: for several such biological networks, the diameter has been studied, for example, in [2,18].

Because of the huge size of real-world networks, in almost all the above cases, the diameter of the strongly connected components has been only estimated. Indeed, most algorithms for finding the exact diameter solve the *all pair shortest path* problem, that is the problem of finding the shortest path between all pairs of nodes of the graph: this can be done either by applying *text-book algorithms* (such as breadth-first searches) for solving, for any node, the *single source shortest path* problem, or by applying fast matrix multiplication algorithms with sub-cubic complexity (see, for example, [20]). However in the context of real-world networks, these approaches are not practical and usually just estimations or bounds can be provided.

To this aim, some algorithms have been proposed in order to estimate the cumulative distribution of the shortest path lengths of a graph and to thus obtain an estimation of the diameter with a small additive error: this is the case, for example, of the algorithms proposed in [22,4,14]. In other works lower bounds for the diameter have been provided by using a *sample* of the nodes and returning their maximum eccentricity, where the eccentricity of a node u is defined as $\max_{v \in V} d(u,v)$ (see, for example, [17]). In the case of *undirected* graphs a lower bound can also be provided by using the so called *double sweep* algorithm, in short 2-Sweep: pick the farthest node from a random node and return its eccentricity. This idea can be iterated by picking at each step the farthest node from the previous one and maintaining the highest found eccentricity (see, for example, [18]). In real-world networks, this lower bound is very good and, in order to prove its effectiveness, several works, like [16,10], propose strategies to find a matching or close upper bound. Recent advances [9,24] have experimentally shown that in real-world networks a matching between a lower and upper bound for the diameter can be found by applying a very small number of computations of breadth-first searches. The most striking result, along this line of research, has been obtained in [1], where the algorithm proposed in [9] has been applied in order to compute the diameter of the biggest connected component of the Facebook network (approximately 721.1M of nodes and 68.7G of edges). In the case of *directed* graphs, in order to obtain a lower bound for the diameter, the idea of the *double sweep* has been adapted by [6]: pick the farthest node from a random node and return its backward eccentricity, i.e. its eccentricity in the graph with reversed arcs (we will make use of this adaptation in this paper).

In this paper we generalize the idea of the algorithm proposed in [9], by presenting the directed *i*FUB (in short, D*i*FUB) algorithm, in order to calculate the diameter of the strongly connected components of directed graphs. As far as we know, D*i*FUB is the first algorithm which is able to compute exactly the diameter of the strongly connected components of huge real-world directed graphs. The D*i*FUB algorithm can also return a pair of nodes whose distance is exactly equal to the diameter, and a natural adaptation of it works also for weighted graphs.