Hub Labels

Sergey Hayrapetyan

February 18, 2016

Abstract

Point to point distance computation is a fundamental problem in a graph theory. Dijkstras algorithm solves this problem in near linear time. But it observes insignificant nodes to get the correct answer. An alternative solution for point to point shortest path distance computation is using Hub-Labels. Main idea is to save for each node of the graph important nodes for itself and process very fast shortest path queries in the graph. The label for each node contains hubs with its distance from and to the node. It has excellent query performance, which makes the ability to calculate the distance between 2 nodes in the graph in very small time. The major disadvantage is the storage expenses, as it requires to save Labels for each node of the graph.

1 Introduction

Hub labeling (HL) is a powerful technique to calculate the distance between two points of the graph with an excellent performance in time. Hub labels gives the advantage to observe only most important nodes of the graph during the query between the source and the target nodes. At first we need to calculate the most important nodes for each node of the graph and save its distance from and to the nodes with its ID. That means that each nodes of the graph has two Labels: forward and backward labels.

Forward label contains important nodes with distance from the node and to its ID, i.e. L_f(v) forward label of node v has elements $\{(ID_1$, distance(v, ID_1)), (ID_n, distance(v, ID_n))$\}$. The same is for backward label in contrast to the distances, where the saved distance in label is from the hub to the node (ID_n, distance(ID_n, v)). After preprocessing the graph and calculating labels for each node of the graph, we are able to make fast queries to find distance between random 2 nodes of the graph.

To calculate the distance between s and t nodes, the algorithm uses only forward labels of s and backward labels of t.

It is preferable to have hubs sorted in labels by its ids for faster queries. A very important condition is that for all nodes of the graph, there should be at least one common node, which covers the shortest path P_{st} . This is called Cover Property.

$$\forall \text{ s-t shortest paths } \exists \text{ v} \in L_f(\text{s}) \cap L_b(\text{t}) \text{ s.t. dist(s,v) + dist(v,t)} = minP_{st};$$

The size of the label is the number of hubs. The label size should be as small as possilble to save memory space and process faster queries. That means that the algorithm which generates labels should not only satisfy the cover property but also it should generate labels with small sizes. In the first section is described about hub labels and its advantages. In the second section we will talk about hierarchical and canonical labels. Afterwards in the next section is described the way of label generation, included queries and label pruning by calculating the canonical labels. In last 2 Chapters it will be explained how to change IDs of hubs which are appearing frequently in labels to compress it and save space. The last section is about results, label sizes and query time.

2 Canonical Labeling

We say that a hierarchical labeling L respects a total order r if the implied partial order is consistent with r.

Given an order r, a canonical labeling is the labeling that contains only hubs which are the highest ranked on the shortest path P_{st} and are included in Forward label of s and in backward label of t. A canonical labeling is hierarchical by construction: the vertex v that we add to the labels of s and t has the highest rank on P_{st}, and therefore r(v) $\geq$ r(s) and r(v) $\geq$ r(t).

This also implies that the canonical labeling respects r.

Lemma 1. *Let $\mathcal{L}$ be a hierarchical labeling. Then the set of vertices on any shortest path has a unique maximum element with respect to the partial order implied by $\mathcal{L}$.*

Proof. The proof is by induction on the number of vertices on the path. The result is trivial for paths with a single vertex. Consider a path $v_1, \ldots, v_k$ with k $\geq$2. The subpath $v_2, \ldots, v_k$ has a maximum vertex v_i by the inductive assumption. Consider the subpath $v_1, \ldots, v_i$. The internal vertices v_j are note in $L_r(v_i)$ by the choice of v_i. Therefore either $v_i \in L_f(v_1)$ (and v_i is the maximum vertex on the path), or $v_1 \in L_r(v_i)$ (and v_1 is the maximum vertex). □

Theorem 1. *Let $\mathcal{L}$ be a hierarchical labeling, r any total order such that $\mathcal{L}$ respects r, and $\mathcal{L}'$ the canonical labeling for r. Then $\mathcal{L}'$ is contained in $\mathcal{L}$.*

Proof. Consider the shortest path P from s to t, and let v be the maximum rank vertex on P. Let $\mathcal{L} = (L_f\ , L_b)$. We show that v $\in L_f$(s); the case v $\in L_b$(t) is similar. Consider the shortest path P' from s to v. Since shortest paths are unique, P' $\subseteq$ P, and therefore v is the maximum rank vertex on P'. It follows that the only vertex of P that is in L_b(v) is v. Thus v must be in L_f(s). □

From theorem we can consequence that size of Canonical Labels is smaller than the size of Hierarchial Labels. Therefore we can prune hubs in labels by calculating the canonical labels.

Contents

List of Figures

List of Tables