Dynamic Graph Labeling

Andreas Kinell

Instructors:
Regina O'Dell
Roger Wattenhofer

November 6, 2004

Contents

Abstract

The object of this term project (Semesterarbeit) is Dynamic Distance Labeling. A great amount of time has been used to read and understand related papers. Several (small) proofs in different areas have been shown, but are not documented here, since there is no direct relation to this document. While developing an own distance labeling scheme occured to be impossible due to already very good proposals, it was intended to apply an existing labeling scheme to a variation of a Planar Unit Disk Graph. It resulted that the most interesting questions aroused during the last days of the project and still remain to be answered. The main achievement, somewhat unsatisfying, lies therefore in providing directions on which further investigations could be made.

1 Introduction

1.1 Definition of Dynamic Graph Labeling

Using the term *dynamic graph labeling* we refer to *dynamic distance labeling in graphs*. To define dynamic graph labeling, some preliminary definitions are recalled:

- Distance: Given an undirected connected weighted graph G and two nodes u and v, we denote by $d_G(u, v)$ the distance between u and v in G, i.e. the minimum weight of a path between them.
- Node-labeling for graph G: A non-negative integer function L that assigns a label $L(u, G)$ to each node u of G.
- Distance-decoder: A function f which, given two labels λ_1, λ_2, returns an integer $f(\lambda_1, \lambda_2)$. Note that f is independent of G.
- Distance labeling: $\langle L, f\rangle$ is a distance labeling for G if $f(L(u, G), L(v, G)) = d_G(u, v)\ \forall u, v \in V(G)$. i.e. if f given the labels u, v returns the distance between u and v.
- Distance labeling scheme: $\langle L, f\rangle$ is a distance labeling scheme for the graph family $\mathcal{G}$ if it is a distance labeling for every graph $G \in \mathcal{G}$.
- Timestep: A timestep t is an interval of time, such that in any interval, not more than *one* action is performed (i.e. the addition or removal of a node in a graph).

Definition 1.1. *Dynamic Graph*
A dynamic graph is a graph $G \in \mathcal{G}$ at a given timestep t, denoted by $G(t)$ or $G(V, E, t)$, where $V \in G(t)$ and $V' \in G(t + 1)$ may only differ in one element $\forall t \geq 0$.

Note that this definition implies that G(t) $\in \mathcal{G}\ \forall t$.

Definition 1.2. *Dynamic Graph Labeling Scheme*
Let $G(t) \in \mathcal{G}$ be a dynamic graph. A dynamic graph labeling scheme is a distance labeling for all graphs $G(t),\ t \geq 0$.

This is a syntactical redefinition only. It makes sense though, since we want a dynamic graph labeling scheme to fulfill different requirements.

1.2 Objective

The ultimate goal is to achieve a "good" routing algorithm for dynamic graphs. This can be achieved using "good" distance labels. Particularly, we want the labels to be of small size and the decode-function to be fast. When it comes to dynamic graphs though, a great concern is the time needed to construct a label (sub)set for a given graph. That is, the time complexity of recalculating labels in case of addition or removal of a node.

1.3 Our Contribution

In Section 3.1 it is shown by example that in the worst case, an addition or removal of one node in a dynamic general graph requires recalculation of $\Omega(n)$ labels, even for s-approximative schemes, where calculation time of one node is in $\Omega(n)$
A known distance labeling scheme is applied to a Planar Unit Disk Graph. Thus, the properties of the Unit Disk Graph as a network representation still hold while the distance labeling scheme is better than any scheme for normal Unit Disk Graphs. The drawback is a high constant s for the s-approximative scheme.
Also, a short and simple algorithm is shown to keep the number of nodes in a dominating subset of a graph smaller or equal to $n/2$, where n is the number of nodes of the graph.

2 Static Distance Labeling

2.1 Achievements in General Graphs

Concerning maximum label size and time complexity of the decoder function, very good proposals have been made. Peleg and Gavoille proved in [GP01] a lower bound of $\Omega(n)$ for the label size and offered $11n+O(\log n \cdot \log\log n)$ as an upper bound with a decode function time complexity of $O(\log\log n)$. This time complexity is close to constant for real applications. Still, the minimum label size for a constant time complexity may be of interest.

2.2 Achievements in Unit Disk Graphs

As will see later in Section 3.2, some special graphs, like the Unit Disk Graph [CCJ90] are better fitted to model real networks compared to general graphs. Therefore, we try to find some better labeling scheme for this type of graphs. Concretely, we will try to convert a Unit Disk Graph to a planar graph, and then apply an approximate distance labeling scheme for planar graphs. One possible solution is to transform a Unit Disk Graph to a Gabriel Graph. However, this transformation does not guarantee s-approximative distances for a constant s. This means $d_{UDG}(u,v) > s \cdot d_{GG}(u,v)$ in the worst case.
A better approach is given by Yu Wang and Xiang-Yang Li, who proposed a planar spanner for the Unit Disk Graph, such that distances have constant stretch factor both hop-wise and length-wise [WL03].

3 Dynamic Graph Labeling

In the following Section 3.1, we note that for general graphs the cost of adding a node to the graph and updating the existing labels accordingly lies in $\Omega(n^2)$. This means, virtually speaking, the recalculation of the whole label set, every time a new node is added. Thus, no great speedup can be gained by improvements. The solution seems to be a focused view on special graph families. If a certain degree of locality can be defined, inserting a new node does not affect the whole graph and therefore can be done in less time.

3.1 Applying Static Methods to Dynamic Graphs

As we will see, distance labeling is not fitted for dynamic graphs, since the lower bound of creating a label set is in $\Omega(n^2)$.

Claim 3.1. *Any distance labeling scheme on general dynamic graphs needs at least $\Omega(n^2)$ time to create a label in the worst case. This holds even for s-approximative schemes.*

Argumentation. We construct an unweighted graph G as follows:

- Let H_1 be a general graph and H_2 a graph with a diameter $diam_{H_1} \geq diam_{H_2}$
- Let B be a Bridge that connects $x \in H_1$ with $y \in H_2$ such that the distance $d(x, y) \geq 2s(diam_{H_1} + 1)$
- Let $\langle L, f\rangle$ be a distance labeling scheme for the graph family $\mathcal{G}$, $G \in \mathcal{G}$. Let $\mathcal{L}$ be the set of labels for G. f is the distance calculation function (decode-function).

Now, we add a vertex v, such that $\exists w \in H_1 : d(w, v) = 1$ and $\exists z \in H_2 : d(z, v) = 1$. The distance $d(w, z)$ is now obviously 2. $f_{\mathcal{L}}(w, z)$ computes $d(x, y) + d(x, w) + d(y, z)$. As $d(x, y) \geq 2s(diam_{H_1} + 1)$ it is clear that $f_{\mathcal{L}}(w, z) > d(w, z) \cdot s$. In fact, $\forall a \in H_1, b \in H_2$ the distance $d(a, b) = d(a, x) + d(x, y) + d(y, b)$ has changed to $d(a, b) = d(a, w) + 2 + d(z, b) \leq 2 + 2 \cdot diam_{H_1}$. The distance function computes $f_{\mathcal{L}}(a, b) \geq 2s(diam_{H_1} + 3) > 2 + 2 \cdot diam_{H_1} \cdot s \geq d(a, b) \cdot s$, $\forall a \neq x, b \neq y$. Hence, every label of every node $a \in H_1$ has to be recalculated, as well as every label of every node $b \in H_2$.
It remains to be shown, that the calculation of every label needs $\Omega(n)$ time: Clearly, $\exists w \in V(H_1)$ s.t. creation of the label needs $\Omega(n)$ time, since for every node $z \in V(H_2)$ it has to be tested, if the newly added vertex v reduces $d(w, z)$ or not (v may or may not be connected to every $z \in V(H_2)$).
Knowing the label of one $w \in V(H_1)$ does not speed up calculation of $w' \neq w \in V(H_1)$. Suppose we know $d(w', z)\ \forall z \in V(H_2)$ and $d(w, w')$. We then know, that $d(w, z) \leq d(w', z) + d(w', w)\ \forall z \in V(H_2)$. Since v may or may not be connected to any $w' \in V(H_1)$, we still have to test $d(w, w')\ \forall w' \in V(H_1)$.

□

3.2 Network Model

While there are distance labeling schemes for general graphs which allow calculating the distance between two nodes in $O(\log\log n)$ time and labelsize $11n + O(\log n \cdot \log\log n)$, those schemes are not necessarily suited for dynamic graphs since the cost of calculating the label set may be too high. Therefore, the better approach is to use graph families that better approximate a given network model, yet to be defined.
From the mobile computing point of view, a node of a graph is a device and its edges represent the connections to other devices. As the devices usually communicate wireless, they have a transmission radius.
For our network model, we make the following assumptions:

- All nodes are equal (i.e. they only differ in their label).
- Every node has a "transmission radius" r. Since all nodes are equal, we can assume $r = 1$.
- The nodes are distributed on a plane.
- A node has a restricted (constant) maximum of connections
- A node may join or leave the graph. In one timestep, only one node may do so.

The resulting dynamic k-bounded unit disk graphs may be used to model for example a mobile cell phone network. Planar distribution of the nodes of course does not imply a planar graph. The model does however introduce a geometric *locality*.

4 Dynamic Distance Labeling in Special Graphs

4.1 The Planar Unit Disk Graph

[WL03] shows a planar spanner for the Unit Disk Graph using an underlying dominating set of the graph. [GKKP00] shows a 3-approximative distance labeling scheme for planar graphs with $O(n^{1/3})$ bit labels.
We can now construct a planar spanner of a Unit Disk Graph and apply the mentioned distance labeling scheme to it. This results in a s-approximative scheme, since the planar spanner assures a (big) constant hop stretch factor ($< 49 * 64 + 36$).
The main interest lies in the behavior of this scheme in dynamic graphs. The planar spanner can be constructed locally. A node either is a dominatee or a dominator. If a newly inserted node connects to at least one dominator, it becomes itself a dominatee. It cannot connect to more than five dominators (due to the property of the UDG). If it connects to dominatees only, it becomes a dominator. Since the whole spanner can be computed locally, the changes caused by the insertion of the new node also can be computed in a localized manner. Moreover, the construction of the new spanner takes constant time ([WL03], page 2).
For the following notes, the research of the mentioned planar distance labeling scheme is required.

The used distance labeling scheme partitions the graph into regions and subregions. Every node is either an internal node, or it is a boundary node. If the newly inserted node n connects to nodes of one subregion only, it becomes an internal node of that subregion. Its label can then be constructed locally, since $d(n, u) = d(v, u) + 1 \ \forall u \in G$, where v is the neighbor of n with shortest distance to u.
If n connects to several regions/subregions, it becomes a boundary node. Then, naivly, all nodes must check, if some $p_u(R_{i'})$ has changed and all nodes in same subregions as n must check, if some $p_u(S_{j'})$ has changed. This prevents a locally bounded update. There might exist a locally computable solution.

4.2 Dominating Set Algorithm

The following algorithm determines a dominating set S of a graph G of size $|S| \leq n/2$, where n is the number of nodes in G. An application may be a distance labeling scheme with some bound directly depending on the size of the dominating set. Inserting nodes into an existing graph could make the size of the dominating set up to five times as big as the size of the dominatee set. Then, the dominating set eventually has to be recalculated

Algorithm 4.1. *Construction of a Dominating Set S of a graph G*

1. *pick a node $n \in V(G)$ and color it A*
2. *color all uncolored neighbors of all nodes with color A with the color B*
3. *color all uncolored neighbors of all nodes with color B with the color A*
4. *if there are uncolored nodes, goto step 2*
5. *if $|S_A| > |S_B|$ then: $S = S_B$, else: $S = S_A$.*

S_A is the set of nodes with color A, S_B is the set of nodes with color B. The resulting dominating set is not necessarily minimal but fulfills $|S| \leq n/2$.

4.3 Conclusions

There are two points of interest. First, it remains to be shown, if the label of a newly inserted node into the Planar Unit Disk Graph, acting as a boundary node in the proposed distance labeling scheme, can be computed locally. The second point is the big constant s in the resulting s-approximating distance labeling scheme. Wang and Li said, that the constant can be improved by more careful analysis of their spanner.
The exact costs of adding a node to a Planar Unit Disk Graph remain to be calculated. If the answer to the first point is negative, only an amortized analysis makes sense. Also, it is assumed that removing a node is a "reversed" action of inserting a node. It remains to be shown if this holds.

References

[GP01] C. Gavoille, D. Peleg: *Distance Labeling in Graphs*. Proceedings of the twelfth annual ACM-SIAM symposium on Discrete Algorithms, 2001.

[CCJ90] B.N. Clark, C.J. Colbourn, D.S. Johnson: *Discrete Mathematics*. 86(1-3): 165-177 , 1990.

[WL03] Yu Wang, Xiang-Yang Li: *Geometric Spanners for Wireless Ad Hoc Networks*. Proc. 22nd IEEE Int'l Conf. Distributed Computing Systems (ICDCS), 2002.

[GKKP00] C. Gavoille, Michal Katz, Nir A. Katz, Christophe Paul: *Approximate Distance Labeling Schemes*. Proceedings of the 9th Annual European Symposium on Algorithms, 2000.

[KWZ02] F. Kuhn, R. Wattenhofer, A. Zollinger *Geometric Ad-hoc Routing for Unit Disk Graphs and General Cost Models*. Technical Report TR 373, ETH Zurich, June 2002.

[KPR02] A. Korman, D. Peleg and Y. Rodeh *Labeling Schemes for Dynamic Tree Networks*. In Proc. 19 Symp. on Theoretical Aspects of Computer Science, 2002.