Reachability and distance queries via 2-hop labels

Edith Cohen * *Eran Halperin* † *Haim Kaplan* † *Uri Zwick* †

August 27, 2002

Abstract

[1] Reachability and distance queries in graphs are fundamental to numerous applications, ranging from geographic navigation systems to Internet routing. Some of these applications involve huge graphs and yet require fast query answering. We propose a new data structure for representing all distances in a graph. The data structure is *distributed* in the sense that it may be viewed as assigning *labels* to the vertices, such that a query involving vertices u and v may be answered using only the labels of u and v.

Our labels are based on *2-hop covers* of the shortest paths, or of all paths, in a graph. For shortest paths, such a cover is a collection S of shortest paths such that for every two vertices u and v, there is a shortest path from u to v that is a concatenation of *two* paths from S. We describe an efficient algorithm for finding an *almost optimal* 2-hop cover of a given collection of paths. Our approach is general and can be applied to directed or undirected graphs, exact or approximate shortest paths, or to reachability queries.

We study the proposed data structure using a combination of theoretical and experimental means. We implemented our algorithm and checked the size of the resulting data structure on several real-life networks from different application areas. Our experiments show that the total size of the labels is typically not much larger than the network itself, and is usually considerably smaller than an explicit representation of the transitive closure of the network.

1 Introduction

We consider the problem of efficiently answering distance or reachability queries in directed or undirected graphs. We focus on scenarios in which the graph/network is given to us explicitly and we are able to do some preprocessing of it. The generated data structure should be fairly compact and should speed-up query answering by as much as possible. Often, the computational resources available for processing queries are weaker than those available during the preprocessing stage. This is the case, for example, in geographic navigation systems, where the preprocessing may be done on a large computer, while queries are answered using a much weaker processor installed in a car.

There are two naive solutions to the problem. The first is to precompute answers to all possible queries, e.g., by solving the all-pairs shortest paths (APSP) problem, or by computing the transitive closure of the network. Each query could then be answered in constant time. This however, is not a viable option, especially for sparse networks, as the data structure produced for an n-vertex graph would be of size $\Omega(n^2)$. The second is to do no preprocessing at all and answer each query, e.g., using a single-source shortest path (SSSP) computation. The space requirements here are minimal, but answering a query on an m-edge graph may take $\Omega(m)$ time.

We present here a new processing scheme, and a corresponding query answering algorithm, for answering reachability and distance queries. Our preprocessing scheme generates a data structure of reasonable size, i.e., not much larger than the original network, and typically much smaller than, say, an explicit representation of the transitive closure of the network. Given the precomputed data structure, our query answering algorithm can answer distance and reachability queries quickly, much faster than what is possible without preprocessing.

*AT&T Labs–Research, 180 Park Avenue, Florham Park, NJ 07932, USA. E-mail: `edith@research.att.com`.

†School of Computer Science, Faculty of exact sciences, Tel-Aviv University, Tel Aviv 69978, Israel. E-mail: `{heran,haimk,zwick}@post.tau.ac.il`.

[1] Preliminary version of the paper appeared in the proceedings of the ACM/IEEE Symposium on Discrete Algorithms (SODA) 2002.

The data structure generated by our preprocessing algorithm is distributed. We assign to each vertex of the network a *distance* or *reachability* label such that we can later calculate the distance (or reachability relation) between two vertices using only the labels of these two vertices. Distance labels were considered, among others, by Peleg [10], Gavoille *et al.* [4] and Thorup and Zwick [12]. All these papers, however, consider only *undirected graphs*, and only worst case results. We are interested mainly in directed graphs, and in the performance of the proposed labeling scheme on networks that occur in practice.

Our labels are based on the concept of *2-hop covers*. Let $G = (V, E)$ be a (directed or undirected) graph. For every $u, v \in V$, let P_{uv} be a collection of paths from u to v in G. (For example, P_{uv} may be the set of all shortest paths from u to v.) We define a *hop* to be a pair (h, u) where h is a path in G and $u \in V$, the *handle* of the hop, is one of the endpoints of h. We say that a collection of hops H is a 2-hop cover of the collection $P = \cup_{uv} P_{uv}$, if for every $u, v \in V$, if $P_{uv} \neq \emptyset$, then there is a path $p \in P_{uv}$ and two hops $(h_1, u) \in H$ and $(h_2, v) \in H$ such that p is the concatenation $h_1 h_2$. We show how to obtain a *2-hop labeling* from such a cover by mapping each hop to an item in the label of its handle node.

The nature of each such item and the interpretation of the labels depend on the specific variant of the problem that we want to solve. Consider, for example, undirected distance queries. Each set of paths P_{uv} contains all shortest paths between u and v. The item in the label $L(u)$ of u that corresponds to a hop $(h = (u = v_0, \ldots, v_k), u)$ is $(w(h), v_k)$, where $w(h)$ is the weight of the path h. The distance between two nodes u and v can then be obtained from their labels by taking the minimum, over all nodes that appear in *both* $L(u)$ and $L(v)$, of the sum of the two weights.

The total size of the labels is thus $|H|$ (We count here the *number* of hops in the cover, not their total length.) Each distance/reachability query can then be answered in time that is linear in the size of the corresponding labels. Therefore, on *average*, in $O(|H|/n)$ time.

We show that finding a 2-hop cover (and thus, a 2-hop labeling) of minimum size is an NP-hard problem. We present, however, an efficient and practical algorithm, which we implemented, for obtaining almost optimal 2-hop covers. The size of the 2-hop cover returned by this algorithm is larger than the minimum possible 2-hop cover by at most a logarithmic factor. In practice, we expect the performance ratio of this algorithm to be much better.

Our algorithm produces an almost optimal 2-hop labeling of *any* input graph. Some graphs may have shorter labelings that are not 2-hop labelings. But, for many interesting families of graphs, such as planar graphs, the optimal 2-hop labelings are almost as short as the optimal labeling. Thus without being specifically designed for any such graph family, our algorithm produces almost optimal labelings. It is also expected, as verified by our experiments, to work well on graphs that are, say, 'mostly' planar, or 'mostly' tree-like, as many practical problem are.

We *conjecture* that any n-vertex, m-edge directed graph has a 2-hop cover, and thus a 2-hop labeling, of total size $\tilde{O}(nm^{1/2})$. We show there exist graphs for which *any* distance or reachability labeling is of size $\Omega(nm^{1/2})$.

A useful property of our approach is that by properly selecting the underlying set of paths, labels can be produced for different variants of the shortest paths problem: reachability or distances, directed or undirected, exact or approximate distances, and for the complete or partial set of vertex-pairs. Even though exact directed distance labeling for all pairs of vertices is the most general variant in the sense that all other variants can be reduced to it, the distinction is important as a less-constrained variant often enjoys a more compact labeling. One such example is the family of directed planar graphs: Reachability or $(1+\epsilon)$-approximate shortest paths have worst-case 2-hop labels of size $O(n \log^2 n)$ (using some slight modification of a construction by Thorup [11]) whereas for exact-distances, there is a known $\Omega(n^{4/3})$ lower-bound for any labeling scheme [4]. It is also not too hard to construct specific graphs which admit more compact labeling for less-constrained variants. Thus, 2-hop labels should always be produced for the least-constrained appropriate variant.

We have made an experimental study of our proposed labeling scheme. We used several real-world networks and obtained promising results, showing that the size of the labels produced is typically significantly smaller than what would have been required for explicit representation.

2 Reachability and distance queries

Let $G = (V, E)$ be a weighted graph (which may be directed or undirected) with $n = |V|$ and $m = |E|$. If $(u, v) \in E$ is an edge, we let $w(u, v)$ be the weight (or cost, or length) attached to that edge. In networks used in applications, the weights attached to the edges are non-negative. Our approach, however, works also when there are edges of negative-weight. We let $\delta(u, v)$ be the *distance* from u to v in the graph, i.e., the smallest weight of a path from u to v in the graph, if one exists, or ∞, otherwise. (We assume that there are no negative weight cycles in the graph so all distances are well defined.)

We would like to preprocess the graph G and obtain a compact representation of it such that given a pair of vertices $u, v \in E$, we could quickly answer the following queries:

reach(u, v)**:** Is there a path from u to v in the graph?

dist(u, v)**:** What is the distance from u to v in the graph? Sometimes, we would be satisfied with $(1 + \epsilon)$-approximate distances.

first-edge(u, v)**:** Which edge emanating from u is a first edge on a (shortest) path from u to v?

path(u, v)**:** Find a (shortest) path from u to v in the graph.

Reachability queries are of course special cases of (directed) distance queries as there is a path from u to v if and only if the distance from u to v is finite. If we are only interested in reachability properties we may assume that the weight of all the edges of the graph is 0. Path queries could be answered using repeated first-edge queries. In some cases, it would be possible to speed-up the processing of repeated first-edge queries so that if a shortest path from u to v contains, say, ℓ edges, then the processing time of the ℓ first-edge queries producing it would require substantially less time then ℓ times the time required by a typical first-edge query.

3 2-hop labels

A possible scheme for achieving the objective set forth above is the following: During the preprocessing stage, we attach to each vertex u of the graph a relatively short label $L(u)$ such that for any two vertices u and v, the two labels $L(u)$ and $L(v)$ would contain enough information to answer the required queries. More formally:

Definition 3.1. (Distance labelings) *A* distance *labeling of a weighted, directed or undirected, graph $G = (V, E)$ is a pair (L, F), where $L : V \to \{0,1\}^*$ and $F : \{0,1\}^* \times \{0,1\}^* \to (\mathbb{R} \cup \{\infty\}) \times (E \cup \{\phi\})$, such that for every $u, v \in V$, if $F(L(u), L(v)) = (d, e)$, then $d = \delta(u, v)$, the distance from u to v in the graph. If $d = \infty$, then $e = \phi$. Otherwise, e is the first edge on a (shortest) path from u to v in the graph. The* total bit-size *of the labeling is $\sum_{v \in V} |L(v)|$, where $|L(v)|$ is the length of $L(v)$. The* maximum label size *is naturally $\max_{v \in V} |L(v)|$. We say that a labeling scheme has* linear complexity *if $F(L(v), L(u))$ can be computed in $O(|L(u)| + |L(v)|)$ time.*

Reachability labelings are defined similarly, and $(1 + \epsilon)$-approximate distance labelings are defined by requiring $\delta(u, v) \le d \le (1 + \epsilon)\delta(u, v)$. We focus on reachability/distance labels of the following form:

Definition 3.2. (2-hop distance labeling) *Let $G = (V, E)$ be a weighted directed graph, A* 2-hop distance labeling *of G assigns to each vertex $v \in V$ a label $L(v) = (L_{in}(v), L_{out}(v))$, such that $L_{in}(v)$ is a collection of pairs $(x, \delta(x, v))$, where $x \in V$, and similarly, $L_{out}(v)$ is a collection of pairs $(x, \delta(v, x))$, where $x \in V$. With a slight abuse of notation, we also consider $L_{in}(v)$ and $L_{out}(v)$ to be subsets of V, and for any two vertices $u, v \in V$, we require that:*

$$\delta(u, v) \;=\; \min_{x \in L_{out}(u) \cap L_{in}(v)} \delta(u, x) + \delta(x, v) \;.$$

The size of the labeling is defined to be $\sum_{v \in V} |L_{in}(v)| + |L_{out}(v)|$.

For approximate distances we relax the requirement to $\min_{x \in L_{out}(u) \cap L_{in}(v)} \delta(u, x) + \delta(x, v) \le (1 + \epsilon)\delta(u, v)$.

For undirected graphs we can simplify the definition: For every vertex v, there is only one collection $L(v)$ of pairs $(x, \delta(x,v))$, such that for every $u, v \in V$ we have $\delta(u,v) = \min_{x \in L(u) \cap L(v)} \delta(u,x) + \delta(x,v)$. The size of the labeling is then $\sum_{v \in V} |L(v)|$.

A slightly more general class of distance labelings is the following:

Definition 3.3. (Steiner 2-hop distance labeling) *A* Steiner 2-hop distance labeling *is a labeling in which $L_{in}(v)$ and $L_{out}(v)$ consists of pairs of the form $(x, d(x,v))$ and $(x, d(v,x))$, respectively, where $x \in X$, $d(x,v), d(v,x) \in \mathbb{R}$, and X is some arbitrary finite set. We require that*

$$\delta(u,v) \;=\; \min_{x \in L_{out}(u) \cap L_{in}(v)} d(u,x) + d(x,v) \;.$$

(A similar definition can be stated for undirected graphs and for approximate distances.)

Note that any 2-hop distance labeling is a Steiner 2-hop labeling, with $X = V$ and $d(x,v) = \delta(x,v)$.

We also use the following definition of 2-hop reachability labels.

Definition 3.4. (2-hop reachability labeling) *Let $G = (V,E)$ be a directed graph. A 2-hop reachability labeling of G assigns to each vertex $v \in V$ a label $L(v) = (L_{in}(v), L_{out}(v))$, such that $L_{in}(v), L_{out}(v) \subseteq V$ and there is a path from every $x \in L_{in}(v)$ to v and from v to every $x \in L_{out}(v)$. Furthermore, for any two vertices $u, v \in V$, we should have:*

$$u \rightsquigarrow v \quad \textit{iff} \quad L_{out}(u) \cap L_{in}(v) \neq \phi \;.$$

The size of the labeling is defined to be $\sum_{v \in V} |L_{in}(v)| + |L_{out}(v)|$.

A *2-hop Steiner reachability labeling* is defined similarly, but we allow $L_{in}(v)$ and $L_{out}(v)$ to contain vertices from an arbitrary finite set X.

Remarks 1) In Definition 3.1, we measure the label size in bits. On the other hand, in Definitions 3.2 and 3.4 we measure the size of 2-hop labels by the total number of hops they contain. 2) Our 2-hop labelings, as defined, do not support **first-edge** queries. But, it is easy to add first edge information to the hops so that they would support such queries.

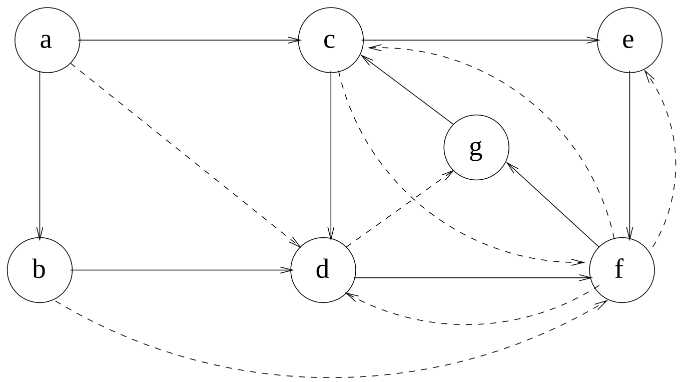
Given the labels $L(u) = (L_{in}(u), L_{out}(u))$ and $L(v) = (L_{in}(v), L_{out}(v))$, we can easily compute $\delta(u,v)$, the distance from u to v in $O(|L_{out}(u)| + |L_{in}(v)|)$ time. (We keep $L_{out}(u)$ and $L_{in}(v)$ in sorted order and merge them to find their intersection.) We can in fact do it also in $O(\min\{|L_{out}(u)|, |L_{in}(v)|\})$ time, if we keep hash tables for the sets $L_{out}(u)$ and $L_{in}(v)$. As an example for 2-hop reachability labeling, consider the graph in Figure 1 and the 2-hop labeling shown in the table next to it. We assume there that each vertex v is contained, by default, in $L_{in}(v)$ and $L_{out}(v)$.

4 2-hop covers

Closely related to 2-hop labelings is the notion of a 2-hop cover of a collection of paths in a graph.

Definition 4.1. (2-hop cover) *Let $G = (V,E)$ be a graph. For every $u, v \in V$, let P_{uv} be a collection of paths from u to v (for undirected graphs we have $P_{uv} \equiv P_{vu}$). Let $P = \{P_{uv}\}$. We define a* hop *to be a pair (h,u), where h is a path in G and $u \in V$ is one of the endpoints of h. We refer to u as the* handle *of the hop. A collection of hops H is said to be a 2-hop cover of P if for every $u, v \in V$ such that $P_{uv} \neq \phi$, there is a path $p \in P_{uv}$, and two hops $(h_1, u) \in H$ and $(h_2, v) \in H$, such that $p = h_1 h_2$, i.e., p is the concatenation of h_1 and h_2. The size of the cover is $|H|$, the number of hops in H.*

Intuitively, a path p can be covered by two hops if p is a concatenation of the two hop paths *and* the handles of the hops are the endpoints of p. We obtain 2-hop labels from 2-hop covers by mapping a hop with handle v to an item in the label of the node v. Thus, the size of this labeling is equal to the size of the 2-hop cover.



v	$L_{in}(v)$	$L_{out}(v)$
a		b, c, d
b		f, d
c	f	d, f
d	f, c	g, f
e	c, f	f
f	d, c	g, c, e
g	d, f	c

Figure 1: Solid edges are the edges of the graph. Dashed edges are hops that are not edges of the graph.

Specifically, we can obtain reachability labels from a 2-hop cover H of the set of *all paths* in G by setting $L_{in}(v) = \{x \mid ((x \leadsto v), v) \in H\}$ and $L_{out}(v) = \{x \mid ((v \leadsto x), v) \in H\}$. The converse also holds: From reachability labels of G we can obtain a 2-hop cover of the same size by adding to H a hop constituting of an arbitrary path $v \leadsto x$ and endpoint v if $x \in L_{out}(v)$ and a hop constituting of an arbitrary path $x \leadsto v$ and endpoint v if $x \in L_{in}(v)$. Similarly we can obtain 2-hop distance labels from a 2-hop cover of the set of *all shortest path* in G, and vice versa. We could also obtain approximate distance labels from a 2-hop cover of the set of all approximate shortest paths and vice versa. Thus we obtain that the size of an optimal 2-hop cover for the set of all paths, the set of all shortest paths, and the set of all approximate shortest paths is equal to the minimum size of a 2-hop reachability labeling, two hop distance labeling, and 2-hop approximate distance labelings, respectively.

The common property of the set of all paths, the set of all shortest paths, and the set of all approximate shortest paths, that makes their 2-hop covers correspond to 2-hop labelings is the following:

DEFINITION 4.2. (HOP INVARIANCE) *A set of paths $P = \{P_{uv}\}$ in a graph $G = (V, E)$ is said to be* hop invariant, *if there exists a collection Q of paths such that*

- *For any two vertices v_i and v_j, there is at most one path $q_{ij} \in Q$ such that $v_i \leadsto^{q_{ij}} v_j$.*
- *Whenever $v_i \leadsto^{p_1} v_j \leadsto^{p_2} v_k$ is a path of P, then q_{ij} and q_{jk} exist and $q_{ij}q_{jk} \in P$.*

LEMMA 4.1. *A set P consisting of all shortest paths between a particular set of vertex pairs (in directed or undirected graphs) is 2-hop invariant. In particular the set of all shortest paths is hop invariant. A similar statement holds for $(1+\epsilon)$-approximate shortest paths, and for the set of all directed paths.*

Proof: In all these cases, it is sufficient to place in Q an arbitrary *shortest* path connecting each pair of nodes. □

THEOREM 4.1. *Finding a minimum 2-hop cover of a collection P of shortest paths in a directed graph is an NP-hard problem.*

Proof: We reduce 3-SAT to the decision version of the minimum 2-hop cover problem of shortest paths in directed graphs. Let S be an instance of 3-SAT with variables x_1, x_2, x_3, …, x_n, and clauses C_1, C_2, …, C_m. We build an instance for the 2-hop cover problem as follows. The graph G consists of a pivot vertex p together with a collection of variable gadgets and clause gadgets. The variable gadget corresponding to a variable x consists of

1. Six vertices a_1^x, a_2^x, b^x, $b^{\overline{x}}$, e_1^x, and e_2^x.
2. Eight arcs, (a_1^x, b^x), $(a_1^x, b^{\overline{x}})$, (a_2^x, b^x), $(a_2^x, b^{\overline{x}})$, (b^x, e_1^x), $(b^{\overline{x}}, e_1^x)$, (b^x, e_2^x), and $(b^{\overline{x}}, e_2^x)$.

We connect the pivot p to the variable gadget by adding the arcs (p, b^x), and $(p, b^{\overline{x}})$.

Let C be a clause $C = (X, Y, Z)$, where X, Y, and Z, are literals of the variables x, y, and z, respectively. For each such clause C we add a vertex c and the arcs (b^X, c), (b^Y, c), and (b^Z, c) (i.e. if $X = x$ we add the arc (b^x, c) and if $X = \overline{x}$ we add the arc $(b^{\overline{x}}, c)$ and similarly for y and z).

For every variable x we define $P_{a_i^x e_j^x}$, $i, j \in \{1, 2\}$, to be the set of directed paths from a_i^x to e_j^x. We also define $P_{pe_j^x}$ to consists of the directed paths from p to e_j^x, $j \in \{1, 2\}$. We call the sets $P_{a_i^x e_j^x}$ and $P_{pe_j^x}$ the *variable paths* that corresponds to variable x. For a clause $C = (X, Y, Z)$ we define $P_{b^X c}$, $P_{b^Y c}$, and $P_{b^Z c}$, to be the set containing the directed path of a single edge from b^X to c, from b^Y to c, and from b^Z to c, respectively. We also define P_{pc} to consists of the three directed paths from p to c. We define every P_{uv} not mentioned above to be empty.

We claim that the 3-SAT instance S is satisfiable if and only if the 2-hops cover instance has a solution of size at most $5n + 3m$.

The proof of this claim is as follows. Assume S is satisfiable and let f be a satisfying assignment. We define a 2-hop cover H as follows. For every variable x if $f(x) = 1$ we add the hops $((a_1^x, b^x), a_1^x)$, $((a_2^x, b^x), a_2^x)$, $((p, b^x), p)$, $((b^x, e_1^x), e_1^x)$, and $((b^x, e_2^x), e_2^x)$ to H. Otherwise we add the hops $((a_1^x, b^{\overline{x}}), a_1^x)$, $((a_2^x, b^{\overline{x}}), a_2^x)$, $((p, b^{\overline{x}}), p)$, $((b^{\overline{x}}, e_1^x), e_1^x)$, and $((b^{\overline{x}}, e_2^x), e_2^x)$ to H. For each clause $C = (X, Y, Z)$ we add the hops $((b^X, c), c)$, $((b^Y, c), c)$, and $((b^Z, c), c)$ to H. It is straightforward to check that if f is satisfying then H is indeed a cover. The size of H is $5n + 3m$.

For the other direction, let H be a 2-hop cover of size $5n + 3m$. To cover the sets $P_{b^x c}$ and $P_{b^{\overline{x}} c}$ for all variables x and clauses C, the cover H must include $3m$ distinct hops of the form (b^X, c), for every clause C and literal $X \in C$. To cover the variable paths corresponding to x we need at least 5 hops consisting of arcs incident with vertices from the variable gadget corresponding to x, and p. Since H is a 2-hop cover of size $5n + 3m$ we cover the variable paths of each variable with exactly 5 hops. It is easy to check that the only way to cover the variable paths with 5 hops is either to include the hops $((a_1^x, b^x), a_1^x)$, $((a_2^x, b^x), a_2^x)$, $((p, b^x), p)$, $((b^x, e_1^x), e_1^x)$, and $((b^x, e_2^x), e_2^x)$ in H or to include the hops $((a_1^x, b^{\overline{x}}), a_1^x)$, $((a_2^x, b^{\overline{x}}), a_2^x)$, $((p, b^{\overline{x}}), p)$, $((b^{\overline{x}}, e_1^x), e_1^x)$ and $((b^{\overline{x}}, e_2^x), e_2^x)$ in H. We define a satisfying assignment to S by setting $f(x) = 1$ if and only if the former set of hops is in H. Since H also covers the sets P_{pc} for all clauses C it follows that S must be a satisfying truth assignment. □

We can, however, efficiently find an almost optimal 2-hop cover if the collection of paths P is hop invariant.

THEOREM 4.2. *Let $G = (V, E)$ be a graph with $|V| = n$, and let P be a hop invariant set of paths in G. Then, there is an efficient algorithm for finding a 2-hop cover of P whose size is larger than the smallest such cover by at most an $O(\log n)$ factor.*

Before proving the theorem we introduce some notation. As before, let P_{uv}, for $u, v \in V$, be the paths of P that start at u and end at v. Let $B_{uv} \subseteq V$ be the set of vertices that appear on paths from P_{uv}. (Note that $u, v \in B_{uv}$, if $P_{uv} \neq \phi$.) A moment of reflection shows that the size of the minimum 2-hop cover of P depends only on the sets B_{uv}, for $u, v \in V$. We also need to define the *densest subgraph* problem and state some results that are known for it.

DEFINITION 4.3. (DENSEST SUBGRAPH) *The* densest subgraph *problem is defined as follows: Given an undirected graph $G = (V, E)$, find a subset $S \subseteq V$ for which the average degree in the subgraph induced by S is maximized, i.e., a set that maximizes the ratio $|E(S)|/|S|$, where $E(S)$ is the set of edges connecting two vertices of S.*

The densest subgraph problem can be solved exactly in polynomial time using flow techniques. One such algorithm is given by Lawler [8, Chapter 4]. The currently best available time bound for the problem is $O(mn \log(n^2/m))$, due to Gallo, Grigoriadis, and Tarjan [3]. It is obtained by reducing the densest subgraph problem to a parametric min-cut problem and then solving it using a parametric max-flow algorithm whose running time is the same as the running time of the non-parametric max flow algorithm of Goldberg and Tarjan [5].

Of more practical interest is a much simpler linear time 2-approximation algorithm for the densest subgraph problem which is a slight modification of an algorithm mentioned by Kortsarz and Peleg [7]. This algorithm iteratively removes a vertex of minimum degree from the graph. This generates a sequence of n subgraphs of the original graph. The algorithm returns the densest of these subgraphs. It is not difficult to check that this algorithm can be implemented to run in linear time, and that it is a 2-approximation algorithm for the densest subgraph problem, i.e., the average degree in the subgraph returned is at least a half of the average degree in the densest possible subgraph.

We can now present a proof of Theorem 4.2.

Proof: **(of Theorem 4.2)** We cast the problem of finding a minimum 2-hop cover of P as a minimum set cover problem. We then apply the greedy algorithm and find a cover that is larger than the optimal cover by at most a logarithmic factor (Chvátal [1], Johnson [6], Lovász [9]). One difficulty which arises is that the resulting set cover instance is huge. We show, however, that it is possible to apply the greedy algorithm to this set cover instance *without* generating it explicitly.

We first recall the flow of the greedy algorithm of the set cover problem. The instance of the set cover problem is a ground set T, and a set $\mathcal{S}$ of subsets of T. For each $S \in \mathcal{S}$, there is an associated weight $w(S)$. The goal is to find a subset $\mathcal{U} \subseteq \mathcal{S}$, such that $\cup_{S\in\mathcal{U}} S = T$, and $\sum_{S\in\mathcal{U}} w(S_i)$ is minimized. The greedy algorithm for the problem is the following. We maintain the set of uncovered elements T', which is initialized to $T' = T$. In each iteration of the algorithm, we add to $\mathcal{U}$ a set S, which maximize the ratio $\frac{|S\cap T'|}{w(S)}$. We iterate until $T' = \phi$.

The set cover instance corresponding to the 2-hop cover instance is constructed as follows. The ground set of elements to be covered is $T = \{(u,v) \mid P_{uv} \neq \phi\}$. For each vertex $w \in V$ and two subsets $C_{in}, C_{out} \subseteq V$, we have a set

$$S(C_{in}, w, C_{out}) \;=\; \{\, (u,v) \in T \mid u \in C_{in}\,,\, v \in C_{out}\,,\, w \in B_{uv}\,\}\,.$$

The weight attached to this set is $|C_{in}| + |C_{out}|$. The vertex w is called the *center* of the set $S(C_{in}, w, C_{out})$. The goal is to find a collection of such sets of minimum total weight that cover T. (Note that the collection of sets is *exponential* in size.)

The proof that this set cover instance is equivalent to the 2-hop cover problem is as follows: If H is a 2-hop directed cover of minimum size, we let $C_{in}(w) = \{u|((u,w),u) \in H\}$ and $C_{out}(w) = \{u|((w,u),u) \in H\}$. We can then cover the set T using the sets $S(C_{in}(w), w, C_{out}(w))$ for $w \in V$.

Conversely, we first claim that we may assume w.l.o.g. that the minimum cover contains at most one set with a given center. If a cover contains two sets with the same center, i.e., $S(C'_{in}, w, C'_{out})$ and $S(C''_{in}, w, C''_{out})$, then these two sets can be replaced, without increasing the size of the cover, by the set $S(C'_{in} \cup C''_{in}, w, C'_{out} \cup C''_{out})$.

Thus, Let $S(C_{in}(w), w, C_{out}(w))$ be the set corresponding to $w \in V$. We can now define a corresponding 2-hop cover, in the following way: $((u,w),u) \in H$ if and only if $u \in C_{in}(w)$ and $((w,u),u) \in H$ if and only if $u \in C_{out}(w)$.

To see that this is indeed a 2-hop cover, consider a pair (u,v) such that $P_{uv} \neq \emptyset$. From the construction of the set cover instance, there exists $w \in B_{uv}$ such that $u \in C_{in}(w)$ and $v \in C_{out}(w)$ and thus the hops $((u,w),u)$ and $((w,v),v)$ are in H.

We next have to show that we can efficiently apply the greedy algorithm to this exponential size set cover instance. Let T' be the part of T that is still uncovered. Initially $T' \leftarrow T$. In each step of the greedy algorithm we are supposed to find a set S with the best ratio $\frac{|S\cap T'|}{w(S)}$. In the directed case we are looking for a set $S(C_{in}, w, C_{out})$ for which the ratio

$$\frac{|S(C_{in}, w, C_{out}) \cap T'|}{|C_{in}| + |C_{out}|}$$

is maximized.

To do that we find, for every $w \in V$, the set $S(w)$ which maximizes the above ratio over all the sets $S(C_{in}, w, C_{out})$, in which w is their center. We construct an auxiliary undirected bipartite graph $G_w = (V_w, E_w)$, which we call *the center graph of* w in the following way. The vertex set V_w contains two vertices v_{in} and v_{out} for each vertex v of the original graph G. We have the undirected edge $(u_{out}, v_{in}) \in E_w$ if and only if $(u,v) \in T'$ and $w \in B_{uv}$. Many of the vertices in G_w may be isolated and can therefore be removed from the graph. It is straightforward to prove that the problem of finding the sets C_{in} and C_{out} that maximize the ratio $|S(C_{in}, w, C_{out}) \cap T'|/(|C_{in}| + |C_{out}|)$ is exactly the problem of finding the *densest subgraph* of G_w.

We solve this problem, computing $S(w)$ for each center $w \in V$, and finally choose the vertex w for which $S(w)$ has the best ratio. We then add the corresponding set $S(w)$ to the cover, update T', and repeat until T' is empty. It is shown by Johnson [6], Lovász [9] and Chvátal [1] that the greedy heuristic achieves a performance ratio of H_t for the set cover problem, where t is the number of elements to be covered and H_t is the Harmonic number. For our problem, the number of elements is equal to the number of vertex-pairs such that there is at least one path in P between them. Thus, we obtain an approximation ratio of

$$H_{|\{ij|P_{ij}\neq\emptyset\}|} \leq H_{n^2} < 2\log n + O(1) \ .$$

This construction is slightly different for undirected paths. The set T to be covered is a set of (unordered) vertex-pairs $\{u, v\}$ such that $B_{uv} \neq \emptyset$. For each vertex $w \in V$ and a subset $C \subseteq V$, we have a set $S(C, w) = \{\{u, v\} \mid u \in C, v \in C, w \in B_{uv}\}$. The proof that this set cover problem is equivalent to our original 2-hop cover problem is essentially as for the directed case: Let $S(C(w), w)$ be the set corresponding to w then $((u, w), w) \in H$ iff $u \in C(w)$. To solve the set cover instance we are interested in $S(C, w)$ which maximizes $\frac{|S(C,w)\cap T'|}{|C|}$. We again first solve this maximization problem separately for each w. Here the auxiliary graph $G_w = (V, E_w)$ contains a single copy of each vertex of V and generally is not bipartite. We have the edge $(u, v) \in E_w$ if and only if $(u, v) \in T'$ and $w \in B_{uv}$. Similarly, the problem of finding the set $S(w) = (C, w)$ that maximize the ratio $|S(C, w) \cap T'|/|C|$ is then exactly the problem of finding the *densest subgraph* of G_w. $\square$

Our approximation algorithm has similar ingredients to an approximation algorithm given by Kortsarz and Peleg [7] for the 2-spanner problem.

4.1 A simple example

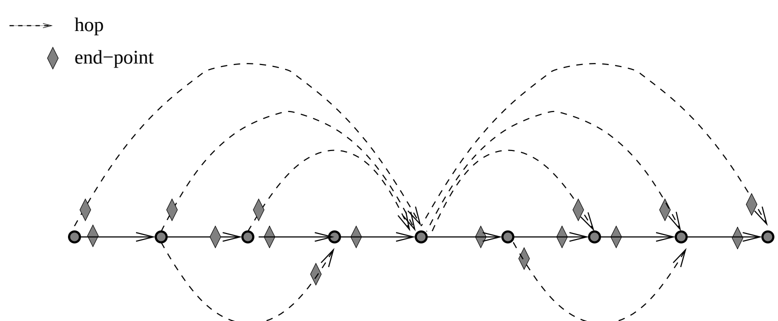


Figure 2: 2-hop cover of a path. Hops are dashed lines or original edges. The diamonds indicate which endpoint is the handle of the hop.

To better visualize the working of our algorithm, we demonstrate its operation for the special case where the input is a graph that consists of single directed path. We name the nodes on the path by consecutive integers $(1, 2, 3, \ldots, n)$. Consider the center graph, G_j of vertex j. For every i, k such that $i < j$ and $k > j$ there is an edge from i_{out} to k_{in} in G_j that corresponds to the path from i to k going through j. The density of the induced subgraph by x hops $l \rightsquigarrow j$ such that $l < j$ and y hops $j \rightsquigarrow l$ such that $l > j$ is $\frac{xy}{x+y}$. For x and y such that their sum is at most $n - 1$ This is maximized if we take $x = y = \frac{n-1}{2}$. So the first densest subgraph that our algorithm will pick would be the whole center graph that corresponds to the node $n/2$. Following this choice, all shortest paths traversing the center are covered and we are left with two independent subproblems: the paths induced by the first half of the original path (up to and excluding the middle node) and the paths induced by the second half of the original path. By the same argument the algorithm will continue and pick the center graph of the middle node in each of the subproblems. It is easy to see that the total number of hops selected by the algorithm is $O(n \log n)$.

Note that if we consider the corresponding 2-hop labels then the maximum label of a node (number of hops in which the node constitutes the marked endpoint) is $O(\log n)$. Observe that even though the algorithm tends to select large center graphs, the corresponding labels are stored only at the "outer" nodes. Thus, it tends to perform well also according to the maximum label metric.

The algorithm works similarly on trees, iteratively selecting as a center the "center of gravity" of the subtree. Thus, generating labeling of size $O(n \log n)$ with maximum label of size $O(\log n)$.

4.2 Specific graph families

We had seen that our algorithm generates 2-hop labeling of size $O(n \log n)$ for all trees (for any variant of the problem). This matches a lower bound of $\Omega(n \log^2 n)$ on the *bit-size* of any tree labeling, due to Gavoille *et al.* [4].

Some other graph families are known to have compact 2-hop labeling: Graphs with separator-decomposition of size $O(n^{\mu})$ have 2-hop labels of size $O(n^{1+\mu} \log n)$ (e.g., see [2]). It is possible that the optimal 2-hop labeling for small-separator graphs is $o(n^{1+\mu} \log n)$. (This would be the case if our Conjecture 5.1 is true.)

For planar graphs, Thorup [11] shows that $O(n \log n)$ size reachability and approximate distance labels are possible. It follows from his construction that there are corresponding 2-hop labels of size $O(n \log^2 n)$. We are not aware of a matching lower bound for 2-hop reachability labelings, thus, it is possible that there are optimal 2-hop reachability labeling of size $O(n \log n)$ for planar graphs. Thorup's result is particularly intriguing since there is an $\Omega(n^{4/3})$ lower bounds on exact distance labels for planar graphs.

Another particular family of interest consists of graphs with bounded degree d where for each pair of vertices there is at least one path between them of length at most $D = \log_d n + o(\log_d n)$ edges. For example a random d-regular graph will have this property with high probability. For such graphs we can obtain 2-hop labelings of size $O(n^{1.5+\epsilon})$ by picking the hops $((v,u),v)$ and $((v,u),u)$ for every two nodes v, u such that there is a path of at most $\lceil D/2 \rceil$ edges from v to u. Observe that the total number of hops is $O(nd^{D/2}) = O(n^{3/2+\epsilon})$.

5 Lower bounds on reachability and distance labelings

In this section we prove lower bounds on the size of reachability labels and 2-hop reachability labels in directed graphs. The bounds we prove also hold for distance labelings in directed and undirected graphs, and it is straightforward to extend the proofs for these cases. We begin with the following simple lemma that gives a lower bound on the size in bits of any reachability labeling (not necessarily a 2-hop reachability labeling).

LEMMA 5.1. *Any reachability labeling scheme must assign some n-vertex m-edge graph reachability labels of total size $\Omega(m \log(n^2/m))$ bits.*

Proof: Consider the set of all directed m-edge graphs on the set of vertices $V = \{1, 2, \ldots, n\}$ in which all edges are directed from $V_1 = \{1, 2, \ldots, n/2\}$ to $V_2 = \{n/2+1, n/2+2, \ldots, n\}$. (Assume that n is even.) There are $\binom{(n/2)^2}{m}$ such graphs, and each one of them has a distinct transitive closure. Hence, no two of these graphs may be assigned reachability labels that are identical for every vertex. It follows that most of these graphs must be assigned reachability labels of total size $\Omega(\log \binom{(n/2)^2}{m}) = \Omega(m \log(n^2/m))$. □

Note that each graph from the family of graphs considered in the proof of Lemma 5.1 may be assigned 2-hop reachability labels of total size $O(m \log n)$ bits. We simply let $L_{out}(v) = \{u \in V_2 \mid (v,u) \in E\}$ for every $v \in V_1$, $L_{in}(v) = \{v\}$ for every $v \in V_2$, and all other sets are empty. Thus, 2-hop reachability labels are almost optimal for this family of graphs. The Corollary below summarizes this discussion and shows that 2-hop labels are almost optimal in the following sense.

COROLLARY 5.1. *Let $g(n, L)$ be the collection of n-node graphs with 2-hop-labels (distance or reachability) of size L ($L \log n$ bits). Then any general labeling scheme assigns labels of maximum size $\Omega(L \log(n^2/m))$ bits on $g(n, L)$.*

Proof: Consider the family of graphs considered in the proof of Lemma 5.1, with n nodes and $L = m$ edges. These graphs have 2-hop-labels of size L. The corollary follows from the proof of the Lemma. $\square$

We next use Lemma 5.1 to obtain a stronger lower bound on the total size of reachability labels.

THEOREM 5.1. *Any reachability labeling scheme must assign to some graphs with n vertices and m edges reachability labels of total size $\Omega(nm^{1/2})$ bits.*

Proof: Consider graphs of the following form: start with a bipartite graph on vertex sets V_1 and V_2, where $|V_1| = |V_2| = m^{1/2}$, with $m/2$ directed edges going from V_1 to V_2. Next, make each vertex of V_1 the center of a star with $n/m^{1/2}$ leaves. All these leaves are disjoint and the edges from these leaves are directed towards the vertices of V_1. Similarly, make each vertex of V_2 the center of a star with $n/m^{1/2}$ leaves. Edges this time are directed away from the vertices of V_2. The total number of vertices in this graph is $2m^{1/2} \cdot n/m^{1/2} + 2m^{1/2} = 2(n+m^{1/2}) = \Theta(n)$, and the number of edges is $2m^{1/2} \cdot n/m^{1/2} + m/2 = m/2 + 2n = \Theta(m)$. (Note that $n \le m \le n^2$ and therefore $m^{1/2} \le n$.)

Let U_i, for $1 \le i \le n/m^{1/2}$, be the set composed of the i-th leaf of every star. For every i, the reachability relation restricted to U_i is isomorphic to the reachability relation on the set $V_1 \cup V_2$. It follows from Lemma 5.1 that for at least one such graph, we need reachability labels of total size at least $\Omega(m)$ bits. Thus, the total size of the labels attached to all the vertices must be at least $\Omega(n/m^{1/2} \cdot m) = \Omega(nm^{1/2})$ bits. $\square$

Our next lemma will allow us to obtain a slightly better lower bound than the one in Theorem 5.1 on the size of *2-hop reachability labelings*. The lemma establishes a lower bound on the size of the optimal 2-hop cover for a set of paths. To specify this lemma we need the following definitions. Given a set P of paths and $v, w \in V$, we define $\overline{h}_v(w)$ to be the number of vertices reachable from v via a path in P going through w. We also define $\underline{h}_v(w)$ to be the number of vertices from which you can reach v through a path in P going through w. Formally, $\overline{h}_v(w) = |\{u \mid w \in B_{vu}\}|$, and $\underline{h}_v(w) = |\{u \mid w \in B_{uv}\}|$. The quantity $\overline{h}_v(w)$ (and similarly $\underline{h}_v(w)$) can be thought of as a bound on the effectiveness of selecting the hop $((w,v),v)$ to our 2-hop cover; $\overline{h}_v(w)$ bounds the maximum number of pairs such that this particular hop can participate in their cover. Similarly, we can bound the maximum effectiveness of the covering of a particular pair, by considering the maximum, over all two hop combinations that can cover it, of the maximum number of pairs that the less-effective of this two hops can participate in covering. Formally, for each pair of nodes (a,b) we define the *efficiency of covering P_{ab}* as follows:

$$\mathrm{eff}(a,b) = \max_{p \in P_{ab}} \max_{v \in p} \min\{\overline{h}_a(v), \underline{h}_b(v)\}$$

LEMMA 5.2. *For any 2-hop cover,*

$$|H| \ge \sum_{(a,b) \mid P_{ab} \neq \emptyset} 1/\mathrm{eff}(a,b) \ .$$

Proof: Any pair (a,b) is eventually covered by 2 hops. One of these hops participates in at most $\mathrm{eff}(a,b)$ paths. Thus, if the cost of each hop is partitioned among the pairs it covers, (a,b)'s share is at least $1/\mathrm{eff}(a,b)$. Thus, the total number of hops is at least $\sum_{ab} 1/\mathrm{eff}(a,b)$. $\square$

We can obtain a slightly stronger lower bound for 2-hop reachability labels than the bound in Theorem 5.1 for general labelings. In Theorem 5.1 the lower bound of $\Omega(nm^{1/2})$ is on the size of the labels in bits whether in the following theorem the lower bound is on the the total number of hops in the 2-hop labels.

THEOREM 5.2. *There exist n-vertex m-edge graphs for which any 2-hop reachability labeling scheme must have a total size of $\Omega(nm^{1/2})$.*

Proof: The graphs we consider are a subset of those used in the proof of Theorem 5.1. We start with *complete* bipartite graph with $|V_1| = |V_2| = m^{1/2}$. As before, we make each vertex of V_1 and V_2 the center of a star with $n/m^{1/2}$ leaves. The proof follows using Lemma 5.2. $\square$

Last, we show that some graphs with large 2-hop reachability labels have much shorter Steiner labels.

Corollary 5.2. *There exists a family of graphs with Steiner reachability labels of size $O(n)$ where the best proper 2-hop reachability labels are of size $\Omega(nm^{1/2})$.*

Proof: The graphs in the proof of Theorem 5.2 can be viewed as 4-layer graphs, with v reachable from u if and only if the layer of u is lower than the layer of v. We use the set $X = \{x_{1,2}, x_{1,3}, x_{1,4}, x_{2,3}, x_{2,4}, x_{3,4}\}$ the identifier $x_{i,j}$ is placed in $L_{out}(v)$ for all v in layer i and in $L_{in}(v)$ for all v in layer j. □

A similar, slightly more involved construction, can be used to generate $O(n)$-size Steiner labels for the undirected uniform-weighted version of this construction:

Corollary 5.3. *There exists a family of undirected graphs with Steiner distance labels of size $O(n)$ where the best proper 2-hop distance labels are of size $\Omega(nm^{1/2})$.*

We conjecture that the bound given in Theorem 5.2 is best possible:

Conjecture 5.1. *Let $G = (V, E)$ be a directed graph with $|V| = n$ and $|E| = m$ and let P the set of all shortest paths in G. Then, there is a 2-hop cover of P of size $O(nm^{1/2})$.*

6 The Pairs-Cover problem

The algorithm in the proof of Theorem 4.2 applies to the following more general problem:

Problem 6.1. (Pairs-Cover) **Input:** *$P_1, \ldots, P_k$, where each P_i is a set of pairs or triples of integers.*
Output: *A set H of pairs of integers of minimum size such that for every $i = 1, \ldots, k$, one of the following holds:*

- *there exists a triple $(c, a, b) \in P_i$ such that $(c, a) \in H$ and $(c, b) \in H$, or*
- *there exists a pair $(c, a) \in P_i$ such that $(c, a) \in H$.*

The problem of computing a 2-hop cover for a hop-invariant set of paths P can be mapped to the following pairs-cover instance: Let $V = \{v_0, \ldots, v_{n-1}\}$ be the set of nodes of G. Each nonempty $P_{v_i v_j}$ is mapped to a different P_ℓ in the pairs-cover instance.

For each v_c that is an interior node on some path in $P_{v_i v_j}$ we append the following pairs/triples to the corresponding P_ℓ: (mapping is slightly different for undirected or directed graphs)

- For directed paths: We include the triple $(c, n+i, j)$ and the pairs (i, j) and $(j, n+i)$ in P_ℓ.
- For undirected paths: We include the triple (c, i, j) and the pairs (i, j) and (j, i) in P_ℓ.

We interpret the output H of the pairs-cover instance as follows: If $(c, i) \in H$ $(i < n)$ then the hop $((v_c, v_i), v_i)$ is placed in the cover. (Recall that since P is hop-invariant, hops can be specified by their endpoints.) For undirected instances hops contain undirected paths. For directed instances hops contain directed paths. We can also have a pair $(c, i) \in H$ with $(i \geq n)$ in which case we rewrite it as $(c, n+a) \in H$ (where $i = n + a$). In this case we place the hop $((v_a, v_c), v_a)$ in the cover.

We now discuss another natural set of problems which this framework covers. We first consider a variant of the 2-hop cover problem where hops do not have handles. Thus, every hop is simply a path, and P_{uv} is covered if there is $P \in P_{uv}$ and two paths $h_1, h_2 \in H$ such that $P = h_1 h_2$. Observe that any solution with handles can be converted to a solution without handles by omitting the handles. Conversely, and solution without handles can be converted to a solution with handles by introducing two hops with both endpoints as handle for every original hop. Thus, the optimal solution to the variant with handles is at most twice that of the variant without handles. Our greedy algorithm can hence be applied to the version without handles, albeit with a loss of another factor of

2 in the approximation ratio. Curiously, it seems that there is no natural modification of the greedy algorithm that obtains the same approximation ratio for the variant without handles.

The variant with handles better fits the labeling application, as there is 1-1 correspondence between 2-hop covers with handles and 2-hop labelings, but variants without handles can have applications.

One such conceivable application is the following problem

Problem 6.2. (Minimum 2-hop Representation) *Let P be an arbitrary collection of strings. A Minimum 2-hop representation of P is a set of substrings H such that every string $s \in P$ is a concatenation $s = h_1 h_2$ of two strings $h_1, h_2 \in H$.*

7 Implementation

Our implementation computes the cover in two phases. The first phase is different for different application (i.e., reachability, approximate or exact, directed or undirected distances), and produces an instance of the pairs-cover problem. The second phase is an approximation algorithm for the pairs-cover problem.

The second phase then constructs a data structure of *center-graphs*: With each number that appears as the first coordinate in some triple we associate a *center*. For each center c we construct an (undirected) *center-graph*. There is a node i in the center graph of c if and only if there is a pair (c, i) or a triple of the form $(c, \bullet, i)$ or $(c, i, \bullet)$ in the input. For each triple (c, i, j) there is an edge (i, j) in the center-graph of c. For each pair (c, i) there is a self-loop on the node i in the center graph c.[2] Figure 3 provides an example of the construction of a center-graph.

The algorithm proceeds in iterations, where the basic operation in each such iteration is *subgraph selection*: We choose a center graph and *select* a subset of the remaining nodes and the corresponding incident remaining edges. The selected nodes are removed. The selected subgraph is in fact an (approximate) densest subgraph in this center-graph. Our implementation of approximate densest subgraph (ADS) is linear in the size (number of edges and nodes) of the center-graph: nodes are maintained sorted by remaining-degree (initially using bucket-sort) and each edge is looked at once.

Each input set P_ℓ is "responsible" for many edges that lie in different center-graphs (each such edge corresponds to a triple or pair in P_ℓ). In our data structure all these edges are *linked* (via doubly linked list). Once one of these edges is *selected* the set P_ℓ is considered *covered* and all these linked edges are removed from their respective center-graphs. Thus, subgraph selection in one center-graph may result in removed edges at other center-graphs.

The main loop guides the selection of the center-graph from which a subgraph selection is made. Centers are maintained in a heap. The key of a center c is the ratio of edges to nodes in the most recently-computed ADS of the center-graph G_c. The heap is initialized through an ADS computation in each center-graph. The main loop repeats the following until all paths are covered (all center-graphs' edges are removed):

- A center c of maximum key is removed from the heap.
- If edges got removed from the center-graph G_c since the last ADS computation was performed, then the ADS is recomputed, and c is placed in the heap with a new key.
- Otherwise, if the center-graph was untouched, the ADS is selected. If unselected edges remain in G_c, then a new ADS is computed, and c is inserted to the heap with a new key.

For this implementation to work correctly, it is important that a center of (approximately) minimum key is selected at each step. To see this, notice that the key of a center-graph (ratio of edges to nodes in densest subgraph) can only decrease if edges are removed from the graph.

[2] We remark that center-graphs are bipartite when the original network is directed but this is generally not true with undirected networks.

We implemented a slightly different variant of the greedy set cover algorithm than the one outlined in the proof of Theorem 4.2: After the algorithm selected several subsets which correspond to the same center: $S(C_1, w), S(C_2, w), \ldots, S(C_k, w)$, the ratio attached to a remaining subset $S(C, w)$ is updated to

$$|S(C,w) \cap T'|/|C \setminus \bigcup_{i=1}^{k} C_i| \ .$$

(instead of $|S(C,w) \cap T'|/|C|$.) where T' is the set of uncovered edges in the center graph G_w. This is achieved by removing nodes that are already selected from the center graph (Note that edges are removed only if both endpoints are selected).

Observe that the numerator of this ratio can decrease as subsets are selected in other centers. The denominator can also decrease but only as a result of a subset being selected from the same center. Even though the ratio associated with a particular subset may increase, it is easy to see that it can never exceed that of the most-recently selected densest subgraph from the same center (if it was, then this subset combined with the recent densest subgraph would have yielded a denser subgraph, hence a contradiction). Thus, the maximum ratio of the densest-subgraph of a center is non-increasing. As a new densest subgraph is computed when a center is placed back in the heap, a center with maximum ratio is still found by our heap implementation.

On our datasets, this variant performed better that the variant where the denominator of the ratio of a subset remains $|C|$. We show that the worst-case approximation ratio is the same as the other variant:

LEMMA 7.1. *This variant achieves a worst-case approximation ratio of* $O(\log n)$.

Proof: For every set $S = (C_S, w)$, let L_S be the elements of C_S that were already chosen in previous rounds of the algorithm. Let $S_1, \ldots, S_k$ be the sets used by our algorithm, in that order. Consider an element v that was covered for the first time by a set S. We let $w_v = \frac{|C_S \setminus L_S|}{|S \cap T'|}$. Notice that

$$\sum_{i=1}^{k} w(S_i) = \sum_{i=1}^{k} |C_{S_i}| \geq \sum_{i=1}^{k} |C_{S_i} \setminus L_{S_i}| = \sum_{v \in T} w_v.$$

Therefore, it is sufficient to show that when v is the j^{th} element covered then $w_v \leq \frac{OPT}{|T|-j}$, where OPT is the value of the optimal solution. Before v was covered, $|T'| \geq |T| - j$. Let $O_1, \ldots, O_{k'}$ be the sets of an optimal cover, in which each two sets have different centers. Clearly, $\sum_{i=1}^{k'} |C_{O_i}| = OPT$. Furthermore, $\sum_{i=1}^{k'} |O_i \cap T'| \geq |T'| \geq |T| - j$. Therefore, there is at least one set O_i, such that

$$\frac{|O_i \cap T'|}{|C_{O_i} \setminus L_{O_i}|} \geq \frac{|O_i \cap T'|}{|C_{O_i}|} \geq \frac{|T| - j}{OPT}.$$

By definition of the greedy algorithm, the set S which covers v will also satisfy that $\frac{|S \cap T'|}{|C_S \setminus L_S|} \geq \frac{|T|-j}{OPT}$, and thus, $w_v \leq \frac{OPT}{|T|-j}$, and the claim follows. □

8 Experiments

We used different synthetic and real networks to evaluate our labeling algorithm. Our datasets selection was guided by two important applications of our labeling scheme, namely routing and geographic navigation systems. For geographic navigation, distance queries can tell a user the time or distance to reach a desired destination. First-edge queries can be used to generate turn-by-turn driving directions.

Routing tables for packets traveling in a communication network constitute another application of distance labels. The model is that each packet carries its destination label, and the current router obtains the next-hop and if desired, the "distance", by considering the label at the current router and the destination label. This is somewhat similar to the way Internet routing is performed now: each packets carries its destination IP-address. The router

network	*# nodes*	*# edges*	*# paths (pairs)*	*label-size*	*compression*
ISP-net	229	880	38466	2969	13.0
BGP	9236	–	164037	22454	7.3
Roads	548	686	128491	6567	19.6
Grid-10	100	180	2744	843	3.3
Grid-20	400	760	42852	5759	7.44
Grid-30	900	1740	212819	18667	11.40

Table 1: Parameters and performance for different networks

network	*# nodes*	*label-size*	*average-label*	*maximum label*	*maximum in/out list*
ISP-net	229	2969	13	29	15
BGP	9236	22454	2.4	145	140
Roads	548	6567	12	28	–
Grid-10	100	843	8.43	15	14
Grid-20	400	5759	14.4	30	30
Grid-30	900	18667	20.7	60	58

Table 2: total label size and the maximum size of the label of a particular node for different networks

looks up the IP-address (longest prefix match) in its local routing table in order to obtain the next-hop. Routing tables, however, are growing in size. As for the structure of the network, they typically have small diameters. For inter-AS[3] graphs, the core of the graph has high expansion whereas intra-AS networks could be almost planar.

ISP-net is the network of a large backbone ISP (Internet Service Provider). The nodes and (directed) edges of the network correspond to routers and links. This particular ISP uses OSPF (Open Shortest Path First) routing, and the edge-weights are the OSPF weights of the links.

BGP is the set of (directed) paths advertised by BGP (Border Gateway Protocol) routers [4]. The nodes of the network are all Autonomous Systems (AS's) in the Internet reachable from the core. The paths are all advertised paths. [5] We considered only paths of 3 or more hops, as shorter paths can be reconstructed by maintaining edge information at each node.

Roads is the road map of Alpine County, CA (USA), obtained from the TIGER census data [13]. This graph is undirected with weights corresponding to actual distances.

Grid-k is a synthetic network. The underlying network is a $k \times k$ grid. Edges are directed in a Manhattan-fashion, with even and odd row-edges directed in opposite directions and similarly, even and odd columns-edges are directed in opposite directions. The edge weights were selected uniformly at random from $[1, 100]$.

Table 1 lists for each network, the number of nodes, number of edges, number of pairs of nodes such that there is a path from one to the other, and the total size of labels (number of hops). The compression ratio is the ratio of the number of pairs to the total size of the labels. The compression ratio varies with different graph sizes and structures and was between 3 to 19.6. Generally, we expect better ratio for larger graphs. In particular, our analysis shows that for the planar k-grid graphs the total size of the labels is $O(k^3)$, whereas explicit representation is $\Theta(k^4)$, thus the ratio is at least $\Omega(k)$.

So far we considered the total size of the labels. Another parameter of interest is the maximum label size of a particular node. This parameter is particularly important for distributed applications. It is also relevant since

[3] AS stands for an autonomous system which is an independent subnetwork of the global network.

[4] This dataset was given to us by Andre Broido from CAIDA

[5] In this case the paths are not necessarily shortest paths and also do not necessarily posses the hop-invariance property. (due to business and other considerations, actual BGP routes do not necessary correspond to shortest paths.) We applied only the second phase of the implementation looking for a 2-hop representation of *every* path in P.

the actual computation of the distance from the labels is linear in the size of the labels. For directed graphs we also consider the maximum size of an in-list or an out-list of a node. The average and maximum label sizes for the different networks are listed in Table 2. Although geared to minimizing the total label size, our algorithm seems to perform well also with respect to the maximum-label metric.

Another interesting question regards the dependence of the label size on the number of covered paths. Our algorithm can be set to stop after any given fraction of the paths is covered (and actually, provides the same performance guarantees even in this case.) The dependence seemed Zipf like and similar across networks (e.g., 20%-25% of the hops suffices to cover half the paths).

9 Concluding remarks

We introduced simple and natural distance and reachability labeling schemes for directed and undirected graphs. Our labelings are derived from 2-hop covers of sets of paths in graphs. We give an efficient algorithm for constructing a 2-hop cover whose size is larger than the smallest 2-hop cover by a factor of at most $O(\log n)$. We *conjecture* that there exists a 2-hop cover of size $\tilde{O}(nm^{1/2})$ of the set of shortest paths in any weighted directed graph with n vertices and m edges. Proving, or disproving, this conjecture, is perhaps the most interesting problem left open. We also demonstrated the effectiveness of our schemes by an experimental analysis using synthetic and real networks from applications such as geographic navigation and Internet routing.

References

[1] V. Chvátal. A greedy heuristic for the set-covering problem. *Math. of Oper. Res.*, 4:233–235, 1979.

[2] E. Cohen. Efficient parallel shortest-paths in digraphs with a separator decomposition. *J. Alg.*, 21:331–357, 1996.

[3] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM J. Comput.*, 18:30–55, 1989.

[4] C. Gavoille, D. Peleg, S. Pérennes, and R. Raz. Distance labeling in graphs. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms, Washington, D.C.*, pages 210–219, 2001.

[5] A. V. Goldberg and R.E. Tarjan. A new approach to the maximum flow problem. *J. Assoc. Comput. Mach.*, 35:921–940, 1988.

[6] D.S. Johnson. Approximation algorithms for combinatorical problems. *J. Comput. System Sci.*, 9:256–278, 1974.

[7] G. Kortsarz and D. Peleg. Generating sparse 2-spanners. *J. Alg.*, 17:222–236, 1994.

[8] E. L. Lawler. *Combinatorial optimization: networks and matroids.* Holt, Reinhart, and Winston, New York, 1976.

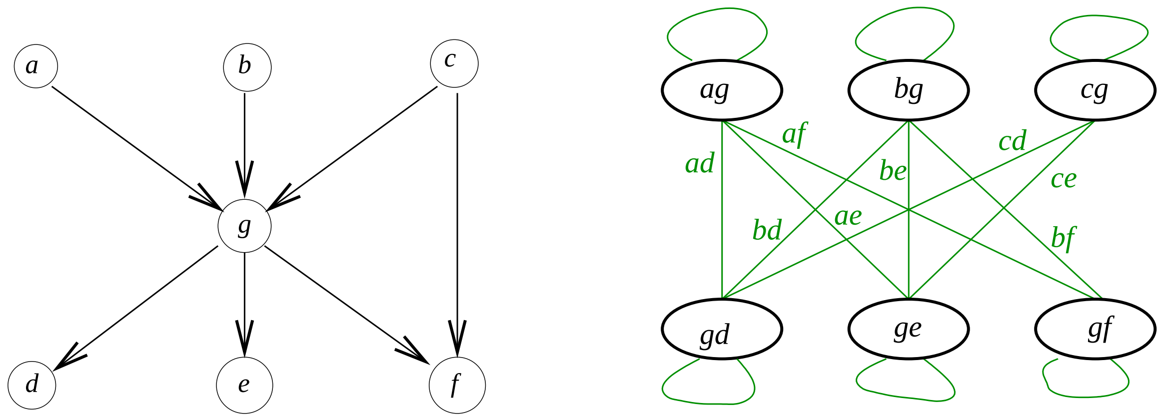
[9] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13:383–390, 1975.

[10] D. Peleg. Proximity-preserving labeling schemes. *Journal of Graph Theory*, 33:167–176, 2000.

[11] M. Thorup. Compact oracles for reachability and approximate distances in planar digraphs. In *Proc. 42nd IEEE Annual Symposium on Foundations of Computer Science*, 2001.

[12] M. Thorup and U. Zwick. Approximate distance oracles. In *Proceedings of the 33th Annual ACM Symposium on Theory of Computing, Crete, Greece*, pages 183–192, 2001.

[13] TIGER. `http://www.census.gov/geo/www/tiger`.



Directed Shortest Paths　　　　Center graph of node "g"

P_{ad}	$=$	{ (a,d-in), (d,a-out), (g,a-out,d-in) }
P_{ag}	$=$	{ (a,g-in), (g,a-out) }
P_{cf}	$=$	{ (c,f-in), (f,c-out) }
P_{gd}	$=$	{ (g,d-in), (d,g-out) }

Some entries of the Pairs-cover instance

Figure 3: This is an example for the construction of a pairs-cover instance and center-graphs for an instance of the directed shortest paths problem on a graph. The graph has 7 nodes labeled $\{a, b, c, d, e, f, g\}$. The finite distances in these graph constitute of 7 ordered pairs with distance 1 and 8 ordered pairs of distance 2. The corresponding pairs-cover instance has a set of pairs/triples for each of these 15 ordered pair of nodes with a finite distance. The table shows some of the entries; other entries are symmetric: The entries for P_{xy} for all shortest paths of length 2 ($ad, ae, af, bd, be, bf, cd, ce$) are symmetric to P_{ad}; P_{bg} and P_{cg} are symmetric to P_{ag}; and P_{ge} and P_{gf} are symmetric to P_{gd}. The figure shows the center-graph for the center g. There are 6 nodes in this center graph, which correspond to shortest-paths for the ordered pairs $\{ag, bg, cg, gd, ge, gf\}$. These hops are denoted by a-out,b-out,c-out,d-in,e-in, and f-in in the pairs-cover instance. The edges of the center-graph are ordered pairs for which the two hops constitute a shortest path; equivalently, xy labels an edge if the triple (g,x-out,y-in) (or (g,y-in,x-out) which is the same) appears in the pairs-cover instance. Self loops correspond to paths that consist of a single hop. The densest subgraph of this particular center-graph is obtained by selecting all nodes; this subgraph thus covers 14 of the shortest paths using 6 hops. Note that P_{cf} (the path from c to f) is not covered by this center-graph, since there is no shortest-paths that traverses the node g. This path can be covered via a single hop (a single node with self loop) in the center-graph of c or the center-graph of f. For this instance, the minimum 2-hop cover constitutes of 7 hops.