Distance oracles in edge-labeled graphs

Francesco Bonchi* Aristides Gionis† Francesco Gullo* Antti Ukkonen†

*Yahoo Labs
Barcelona, Spain
{bonchi,gullo}@yahoo-inc.com

†Helsinki Institute for Information Technology HIIT
Aalto University
{aristides.gionis,antti.ukkonen}@aalto.fi

ABSTRACT

A fundamental operation over edge-labeled graphs is the computation of shortest-path distances subject to a *constraint* on the set of permissible edge labels. Applying exact algorithms for such an operation is not a viable option, especially for massive graphs, or in scenarios where the distance computation is used as a primitive for more complex computations.

In this paper we study the problem of *efficient approximation of shortest-path queries with edge-label constraints*, for which we devise two indexes based on the idea of *landmarks*: distances from all vertices of the graph to a selected subset of landmark vertices are pre-computed and then used at query time to efficiently approximate distance queries. The major challenge to face is that, in principle, an exponential number of constraint label sets needs to be stored for each vertex-landmark pair, which makes the index pre-computation and storage far from trivial. We tackle this challenge from two different perspectives, which lead to indexes with different characteristics: one index is faster and more accurate, but it requires more space than the other.

We extensively evaluate our techniques on real and synthetic datasets, showing that our indexes can efficiently and accurately estimate label-constrained distance queries.

1. INTRODUCTION

Computing *shortest-path distances* between any two vertices of a graph is one of the most fundamental graph primitives, used in a large variety of applications and methods. For today's graph sizes it is often not feasible to compute exact shortest-path distances by relying on some of the well-known basic methods: running the Dijkstra's algorithm, or some of its efficient variants [12], on graphs of billions of vertices may require unaffordable time for a number of real-world applications. This remains true even for moderately-sized graphs if shortest-path queries are used as primitive in the contexts of more complex real-time applications (e.g., in recommender systems). For these reasons, designing fast shortest-path-computation algorithms has become an active research area.

Numerous algorithms have been so far defined for speeding-up shortest-path-distance computation—see [25] for an up-to-date survey. Such algorithms can be broadly classified into *exact* and *approximate*. Although desirable in some scenarios, exact methods usually rely on specific characteristics of the input graph, which may limit the applicability in more general contexts. Instead, approximate methods typically allow for a fine tuning between accuracy and efficiency that makes them able to satisfy the various requirements/constraints of every specific application context.

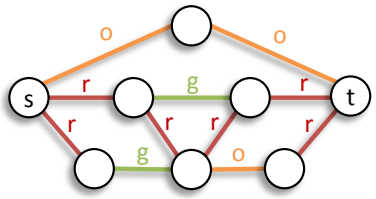

Figure 1: Example edge-labeled graph where the label-constrained shortest-path distance query $\langle s, t, \{r\}\rangle$ returns 4, while the query $\langle s, t, \{r, g\}\rangle$ returns 3, and the query $\langle s, t, \{r, g, o\}\rangle$ returns 2.

Many methods for approximating shortest-path distances are based on the idea of *landmarks* [27, 24, 13, 18, 22, 21, 28]. For a standard graph $G = (V, E)$, the landmark approach works by selecting a subset of vertices $X \subseteq V$, $|X| = k$, and computing the (exact) distances $d(u, x)$ between every pair of vertices $u \in V$ and $x \in X$. At query time, given any two vertices $s, t \in V$, the triangle inequality ensures that

$$\max_{x \in X} |d(x, s) - d(x, t)| \leq d(s, t) \leq \min_{x \in X} (d(x, s) + d(x, t)) .$$

Thus, the shortest-path distance between s and t can be estimated by either the upper bound or the lower bound above, or by some in-between value such as the median [21]. Denoting by n and m the number of vertices and edges in the graph, respectively, the landmark-based index takes $\mathcal{O}(km)$ offline time, as it requires one scan of the graph per landmark, while the storage requirement is $\mathcal{O}(kn)$. At query time, an approximate shortest-path distance $d(s, t)$ is found in $\mathcal{O}(k)$ time by accessing the precomputed landmark-to-vertex distances for s and t.

Shortest paths on edge-labeled graphs. In a variety of application domains it is increasingly common to come across graphs whose edges are associated with a *label* denoting the type of the relationship of the two incident vertices [16, 23, 26, 29, 8, 5]. For instance, users of a *social network* have the possibility of categorizing their own connections in different *social circles* (e.g., "circles" in Google+, or "lists" in Facebook or Twitter) [20], which makes the edges of the social network characterizing different relationship types, such as friends, relatives, colleagues, and so on. RDF resources such as the *Google Knowledge Graph* or *YAGO* are naturally represented as graphs whose links are labeled with the type of the property (predicate) that characterizes the relationship between the two connected entities. In a *co-authorship network* like DBLP

collaborations (links) between any two authors are characterized by the topic(s) of the papers co-authored by those authors [6]. In a *protein-interaction network* edge labels represent different types of interactions between proteins, e.g., physical association, direct interaction, co-localization, and so on [31]. Other examples are *multi-dimensional networks* (i.e., networks derived from the integration of multiple networks) [26, 5], *metabolic networks* [9], *recommendation networks* [19].

When dealing with edge-labeled graphs, it is often required to consider *label-constrained shortest-path distance queries*, i.e., shortest-path distance queries with a constraint on the set of permitted edge labels. More precisely, given two vertices s and t and a set of labels C, the query $\langle s, t, C\rangle$ asks to compute the length of a shortest path from s to t, *using only edges whose label belongs to* C—see Figure 1 for an illustration.

Applications. Label-constrained shortest-path distance queries find natural application in a variety of real-world scenarios. Particularly, being usually a primitive involved in more complex tasks, answering/approximating such queries efficiently is mandatory. As a concrete example, consider some of the today's novel systems for advanced information retrieval and knowledge exploration, such as the Google Knowledge Graph, Facebook Graph Search, or similar RDF resources. As said above, the data format underlying these resources can naturally be represented as a set of entities linked by different types (edge labels) of association. When searching these knowledge resources, one has to answer queries of the form: "How related are entities A and B, contextualized to additional user information C?". Here, C represents the context via which one is interested to associate A and B: it corresponds to the (semantically-annotated) query being issued and/or the interest profile of the current user, and it can naturally be modeled as a set of edge labels. In a real application, the relatedness of A and B is assessed by complex machine-learned ranking functions that exploit various features. Shortest-path distance is a central feature typically considered (it is, e.g., highly-correlated to the prediction of the link between two entities [14]), and, given the presence of context C, the shortest-path distance needs to be label-constrained. Such knowledge-exploration systems need to provide final answers in real-time, thus requiring that all the components of the ranking function, including label-constrained shortest-path distances, must be computed/approximated very quickly.

A similar scenario arises in social networks, where the shortest-path distance is one of the most effective features used for machine-learning-based link-prediction systems [14, 10]. When edge labels are available, the link-prediction systems can be empowered by adding the functionality of predicting the type of the link, and not only whether the link exists or not [1]. To still exploit shortest-path distances as features in the machine-learning core task, both the (offline) model-learning phase and the online prediction phase need to rely on a set of examples of shortest-path distances constrained to the use of permissible labels, for different instances of such constraint label sets. This means that a central task in this context corresponds to answering several label-constrained shortest-path distance queries at a time.

Network alignment, that is the identification of a matching among different (sub-)networks, is a fundamental operation in protein-interaction networks [30]. A specific type of network-alignment query implemented in most existing commercial systems such as *PathBLAST* (`www.pathblast.org`) [17] is: given an input pathway (i.e., a sequence of proteins), find all the pathways of a given target network that match the query pathway. As answering exactly these queries requires solving a number of subgraph isomorphisms, existing systems rely on approximated methods. Such methods can be significantly speeded-up by taking into account the labels naturally present on the edges of any protein-interaction network and exploiting label-constrained shortest-path distance queries. As an example, once having discovered one pathway P that matches the query, the idea is to take the set of labels C lying on the edges of P. Then, when asking if a pathway starting from another protein in a different zone of the network can match the query, one can first run a (approximated) label-constrained shortest-path query specifying C as a constraint label set and use the answer to this query as a pruning rule: if the distance returned is sufficiently larger than the length of P, one can safely conclude that no matching paths can exist starting from the protein being considered. An analogous reasoning clearly holds if simple shortest-path queries are involved, but using label-constrained shortest-path distance queries makes the query more restrictive, thus resulting in more effective pruning.

Challenges. Like non-labeled graphs, answering shortest-path distance queries in edge-labeled graphs efficiently is a very crucial task. Unfortunately, the edge-label constraint creates a non-trivial obstacle to the adaptation of any existing technique for simple shortest-path distance estimation. In fact, there is no way for existing indexes to deal with *the exponential number of possible query constraint label sets*.

For instance, a natural yet naïve way of extending the landmark approach to an edge-labeled graph G with a label set L is to create a different instance of G for each one of the $2^{|L|}$ possible label combinations, and index each graph instance separately. Then, any query including a given constraint label set C is answered from the index of the graph instance corresponding to C.[1] The problem with this naïve approach is of course that the number of graph indexes increases exponentially with the size of the label set L. Even a moderate set of ten labels increases the index size by three orders of magnitude and makes the approach prohibitively expensive.

Outline. In this paper we study the problem of *efficiently approximating point-to-point shortest-path distance queries with edge-label constraints*. To the best of our knowledge, this problem has never been considered so far. Indeed, most research on edge-labeled graphs has focused on the *subset-constrained reachability* problem [16, 29, 8], which is only a special case of the problem we tackle in this paper: those works are only able to say if any two vertices are connected by a path containing only the permissible labels, while we are interested in assessing the distance between the two vertices. To our knowledge, the only work dealing with shortest path in edge-labeled graphs, in particular in road networks, is the one by Rice and Tsotras [23], which is an adaptation of the *contraction hierarchies* [11] method to the context of edge-labeled graphs. However, that work differs from ours in two main aspects. First, it is suited for graphs that have the characteristics of *road networks*: as empirically shown in our experiments, it seems less appropriate for handling other more general graphs (e.g., graphs with a power-law degree distribution, such as social networks). Second, more importantly, Rice and Tsotras focus on exact solutions, while, for all the motivations discussed above, we devise here approximate techniques.

Our goal is to devise indexing strategies to efficiently approximate point-to-point shortest-path distance queries on *real-world* edge-labeled graphs. On such graphs, the number of labels is typically moderate, i.e., in the order of few tens;[2] nevertheless, as al-

[1] An equivalent approach is to index a single graph and store, for each landmark-vertex pair, a distance for each one of the possible $2^{|L|}$ label combinations.

[2] Even on edge-labeled graphs modeling RDF resources, where the number of low-level labels can in principle be much higher, what really matters are the few upper-

ready pointed out above, devising proper solutions even for this number of labels is very challenging, due to the intrinsic exponential blowup that needs to be overcome.

We propose two indexes that adapt the idea of landmarks to edge-labeled graphs in two different ways. The first approach, dubbed Powerset Cover or PowCov for short, is motivated by the observation that certain constraint label sets can be *subsumed* by others. Therefore, it is possible to build the index for a number of minimal, non-redundant label sets that is substantially smaller than $2^{|L|}$. Even though this approach does not provide an asymptotic improvement over the naïve method, it still gives tremendous savings in practice and makes it possible to index shortest-path distances in many real-world networks.

The PowCov index is a valuable solution for most real-world edge-labeled graphs when the number of labels remains within a range of few tens. However, building PowCov indexes becomes unaffordable as the number of edge labels increases beyond this. Thus, we introduce a second approach that keeps a simpler and lighter index. This approach is based on assigning landmarks to a single label, so that they can be used in approximating queries in which their label is part of the constraint label set. We dub this approach Chromatic Landmarks, or ChromLand for short, as each landmark has assigned one "color" (label). While PowCov builds a larger index, ChromLand deals with the inherent complexity of the problem during query processing.

Overall, our indexes offer a tradeoff of indexing time/space vs. accuracy/efficiency: the first index is faster and more accurate, but it has larger space and preprocessing requirements. Also, a key advantage of both our indexes is that the accuracy vs. efficiency trade-off can be fine-tuned by selecting an appropriate number of landmarks: the fewer the landmarks, the faster the query evaluation, and the lower the accuracy.

Contributions. In summary, our contributions are as follows:

- We design two landmark-based index structures for efficiently approximating label-constrained point-to-point shortest-path queries in edge-labeled graphs.
- We propose efficient algorithms for building our indexes. Particularly, for PowCov, we present an efficient way to traverse the label powerset by pruning unpromising label sets.
- We devise novel strategies for finding a good set of landmarks for both the proposed indexes.
- We evaluate our indexes on both real-world and synthetic edge-labeled graphs. Our indexes outperform by a large margin both the exact method and the naïve indexing scheme: they achieve a speed-up factor up to three orders of magnitude compared to exact shortest-path distance computation, while exhibiting small error.

Roadmap. The rest of the paper is organized as follows. In Section 2 we formally state our problem. The two proposed indexes are described in Section 3 and 4, respectively. Section 5 presents experiments, while Section 6 concludes the paper.

2. PROBLEM DEFINITION

The input to our problem is an edge-labeled graph $G = (V, E, L, \ell)$, where V is a set of n vertices, $E \subseteq V \times V$ is a set of m edges, L is a set of labels, and $\ell : E \to L$ is a labeling function that assigns a label in L to each edge in E . For the sake of presentation, we focus on undirected unweighted graphs, even though all our concepts and ideas can easily be extended to handle directed and weighted graphs as well. We also find it intuitive to think of the edge label as the "color" of the edge, so we use the terms *label* and *color* interchangeably in the rest of the paper.

level labels of the hierarchies that are typically exploited to semantically organize the whole set of low-level labels.

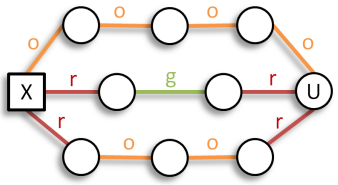

Figure 2: The landmark x and the vertex u are connected by three paths. We can observe that the label sets {o} and {r, g} are SP-minimal with respect to x and u, while {r, o} is not. We thus do not need to compute/store the {r, o}-constrained shortest-path distance, as it can implicitly be derived from the label set {o} that subsumes {r, o}.

Given a set of colors $C \subseteq L$ and two vertices $u, v \in V$, we define the *C-constrained* path $p_C(u, v)$ to be a path between u and v containing only edges e such that $\ell(e) \in C$. The *C-constrained shortest-path distance* $d_C(u, v)$ is the length of a shortest path over all C-constrained paths between u and v. If no such paths exist, we define $d_C(u, v) = \infty$.

In this paper we study *label-constrained point-to-point shortest-path distance queries* (LC-PPSPD), i.e., triples $\langle s, t, C\rangle$, where $s, t \in V$ and $C \subseteq L$, which ask to find the C-constrained shortest-path distance $d_C(s, t)$. Note that the LC-PPSPD problem can be solved exactly in polynomial time by simply removing from G all edges whose color is not in C and computing the shortest-path distance on the resulting graph. Our goal is to devise indexing techniques to perform fast and accurate online approximations.

Note that all our methods are general enough to easily apply to graphs having *multiple labels on the edges*. The only modification needed concerns the definition of C-constrained path, which, in the multiple-label case, is defined as a path with only edges e such that all (or at least one of) the labels of e belong to C. Then, our methods need only trivial adaptations in order to take into account this generalized definition of C-constrained path.

3. POWERSET COVER INDEX

The index we propose here is based on the simple observation that, in real-world graphs, it is likely that different constraint label sets yield the same distances between graph vertices, as shown in the example of Figure 2. Rather than computing and storing distances for all possible label combinations, we thus only consider those ones that are really required.

3.1 Index overview and query processing

We first introduce the notions of *subsumption* and SP-*minimality*. We define these notions for landmark-vertex pairs (x, u), even though they hold for any two vertices $u, v \in V$.

DEFINITION 1. *Given a landmark $x \in X$, a vertex $u \in V$, and two label sets $S, T \subseteq L$, we say that S* subsumes *T with respect to x and u if and only if $S \subseteq T$ and $d_S(x, u) = d_T(x, u)$.* □

DEFINITION 2. *A label set $S \subseteq L$ is said* shortest path-minimal (SP-minimal) *with respect to a landmark $x \in X$ and vertex $u \in V$ if and only if it is not subsumed by any other label set with respect to x and u.* □

The above definitions imply that any non-SP-minimal label set C can be interpreted as "redundant," as the corresponding C-

constrained shortest-path distance $d_C(x,u)$ can be derived by using a subset of C. The idea is better illustrated in the example of Figure 2.

This means that storing all SP-minimal label sets for a vertex-landmark pair (x,u) is sufficient for retrieving the exact distance between x and u, for each subset $C \subseteq L$. The next theorem shows that the SP-minimal label sets are sufficient to retrieve C-constrained shortest-path distances for any landmark-vertex pair (x,u) and any label-set constraint C, as such a distance simply corresponds to the minimum distance taken over all SP-minimal label sets that are subsets of C.

THEOREM 1. *Given a landmark-vertex pair (x,u), let $\mathcal{SP}_{xu}$ be the set of $\langle S, d_S\rangle$ pairs containing all* SP*-minimal label sets S with respect to x and u along with the corresponding S-constrained shortest-path distance d_S. Then, for any label set $C \subseteq L$, the C-constrained distance $d_C(x,u)$ can be retrieved from $\mathcal{SP}_{xu}$ as*

$$d_C(x,u) = \begin{cases} \infty, \text{ if there is no } \langle S, d_S\rangle \in \mathcal{SP}_{xu} \text{ s.t. } S \subseteq C \\ \min\{d_S \mid \langle S, d_S\rangle \in \mathcal{SP}_{xu},\ S \subseteq C\}, \text{ otherwise.} \end{cases}$$

PROOF. By definition of subsumption and SP-minimality, the label sets in $\mathcal{SP}_{xu}$ that account for retrieving the exact distance $d_C(x,u)$ for any label set C are the subsets of C. Hence, if no subset of C is included in $\mathcal{SP}_{xu}$, we can immediately conclude that there are no paths between u and x containing only labels in C, and, therefore, $d_C(x,u) = \infty$. On the other hand, if $\mathcal{SP}_{xu}$ contains at least one subset of C, the distance $d_C(x,u)$ can be retrieved based on the straightforward observation that the S-constrained distance d_S between x and u computed for any subset $S \subseteq C$ represents an upper bound to the distance $d_C(x,u)$. Hence, it holds that $d_C(x,u) \leq d_S$, $\forall \langle S, d_S\rangle \in \mathcal{SP}_{xu}, S \subseteq C$, which clearly implies that $d_C(x,u) \leq \min\{d_S \mid \langle S, d_S\rangle \in \mathcal{SP}_{xu},\ S \subseteq C\}$. However, we recall that $\mathcal{SP}_{xu}$ contains all SP-minimal label sets with respect to x and u. Thus, by definition of SP-minimality, the above inequality must be an equality for some SP-minimal set that subsumes C; this means that $d_C(x,u) = \min\{d_S \mid \langle S, d_S\rangle \in \mathcal{SP}_{xu},\ S \subseteq C\}$. □

Overall, the structure of the PowCov index consists of all sets $\mathcal{SP}_{xu}$, for each landmark-vertex pair (x,u). We partition the label sets S within each $\mathcal{SP}_{xu}$ according to their associated distances d_S and we organize any group of label sets sharing the same distance into a small-redundancy data structure, e.g., a prefix tree.

Query processing. Given a query $\langle s,t,C\rangle$, the C-constrained shortest-path distance between s and t is approximated similarly as in the landmark approach for non-labeled graphs. Indeed, we simply need to retrieve from the index the distances $d_C(x,s)$, $d_C(x,t)$, for all $x \in X$, and approximate $d_C(s,t)$ resorting to the triangle inequality, as described in Section 1. We compute the distances $d_C(x,s)$ (and $d_C(x,t)$) by visiting the groups of label sets in $\mathcal{SP}_{xs}$, starting from the group that has minimum distance (Theorem 1). We stop when a group contains a set S that is subset of C, and we return the corresponding distance d_C. If no subset of C is encountered during the visit, we return ∞.

Complexity. The space complexity of the PowCov index and the query-processing time depend on the size of the various sets $\mathcal{SP}_{xu}$. In particular, if H is the maximum size over all sets $\mathcal{SP}_{xu}$, the total space of the index is $\mathcal{O}(kHn)$, while the query-processing time is $\mathcal{O}(kH|L|)$. In the worst case, H could be $\mathcal{O}(2^{|L|})$. Nevertheless, we remark that H is actually bounded by a function of the maximum *finite* distance $d_{\max}$ in the graph, as shown in Proposition 1.

Algorithm 1 TraversePowerset-BruteForce

Input: an edge-labeled graph $G{=}(V,E,L,\ell)$, a set of landmarks X
Output: for each pair (x,u), where $x \in X$ and $u \in V$, a set $\mathcal{SP}_{xu}$ of $\langle C,d\rangle$ pairs storing all SP-minimal label sets C with respect to x and u along with the corresponding C-constrained shortest path distance d

```
1: SP_xu ← ∅, ∀x ∈ X, u ∈ V
2: for all x ∈ X do
3:     D ← ∅
4:     for all C ⊆ L do
5:         D[C] ← ConstrainedSSSP(G, x, C)
6:     end for
7:     for all C ⊆ L, u ∈ V s.t. D[C,u] < ∞ do
8:         if C is SP-minimal w.r.t. x and u then
9:             SP_xu ← SP_xu ∪ {⟨C, D[C,u]⟩}
10:        end if
11:    end for
12: end for
```

PROPOSITION 1. *It holds that $H \leq \sum_{d=1}^{d_{\max}} \binom{|L|}{d}$, where $d_{\max} = \max_{u,v \in V, C \subseteq L}\{d_C(u,v) \mid d_C(u,v) < \infty\}$.*

PROOF. The claim follows directly from the fact that any SP-minimal label set C cannot have size larger than $d_{\max}$. Indeed, given an SP-minimal label set C with respect to x and u, all labels within C must belong to each C-constrained shortest path between x and u, otherwise, filtering the labels that do not appear in some of these shortest paths out from C would lead to a subset S that subsumes C, thus making C non-SP-minimal. This implies that $|C| \leq d_C(x,u) \leq d_{\max}, \forall C \in \mathcal{SP}_{xu}, \forall \mathcal{SP}_{xu}$. The maximum size H of any $\mathcal{SP}_{xu}$ is therefore not larger than all possible ways of choosing from the input label set L a set of $d \leq d_{\max}$ distinct labels, i.e., $H = \max\{|\mathcal{SP}_{xu}| \mid x \in X, u \in V \setminus \{x\}\} \leq \sum_{d=1}^{d_{\max}} \binom{|L|}{d}$. □

Due to the small-world phenomenon, the distance $d_{\max}$ remains in practice very small. Indeed, as we experimentally show in Section 5, the average number of distances to be stored per landmark-vertex pair is at most quadratic, in most cases even linear, in the number of labels in the input graph.

3.2 Building the index

We now describe how to build a PowCov index. We start with an overview of a basic brute-force approach. Then, we discuss a number of pruning rules to improve its efficiency.

3.2.1 A brute-force algorithm

A brute-force algorithm to build a PowCov index is outlined as Algorithm 1. For each landmark x, it performs the following two main steps. First, a C-constrained single-source shortest path (SSSP) with source x is computed for each label set $C \subseteq L$ (i.e., an SSSP where edges with label not in C are ignored); the result of these SSSPs, i.e., the C-constrained distances $d_C(x,u)$ for all vertices $u \in V$, is stored into the vector $\mathbf{D}$ (Lines 4-6). Note that $\mathbf{D}[C,u] = d_C(x,u)$. Then (Lines 7-11), for each label set $C \subseteq L$ and each vertex $u \in V$, the SP-minimality of C with respect to x and u is checked; if C is recognized as SP-minimal, then it is added (along with the corresponding distance $\mathbf{D}[C,u]$) to the output set $\mathcal{SP}_{xu}$.

The first phase of computing the SSSPs for all landmarks and all label sets takes $\mathcal{O}(2^{|L|}mk)$, while checking SP-minimality for all landmarks, all labelsets, and all vertices takes $\mathcal{O}(2^{|L|}nk|L|)$. The latter is a result based on the following theorem:

THEOREM 2. *Given a landmark $x \in X$ and a vertex $u \in V$, any label set $C \subseteq L$ is* SP*-minimal with respect to x and u if and only if $d_C(x,u) < d_{C'}(x,u)$, for all $C' \subset C$ such that $|C'| = |C| - 1$.*

PROOF. The necessary condition (i.e., C is SP-minimal only if $d_C(x,u) < d_{C'}(x,u)$, for all $C' \subset C$ such that $|C'| = |C| - 1$) easily follows from the definition of SP-minimality: C cannot be SP-minimal with respect to x and u if there exist a subset C' of C such that $d_{C'}(x,u) = d_C(x,u)$.

The sufficient condition (i.e., C is SP-minimal if $d_C(x,u) < d_{C'}(x,u)$, for all $C' \subset C$ s.t. $|C'| = |C| - 1$) holds based on the following observation. If the distance $d_C(x,u) < d_{C'}(x,u)$, for all subsets C' of C of size $|C'| = |C| - 1$, then this must hold for *any* subset of C as well, regardless of the size. This implies that C is SP-minimal. □

That is, the SP-minimality of a label set C is checked in $\mathcal{O}(|C|) = \mathcal{O}(|L|)$ time by considering only the set of (previously computed) distances $\{\mathbf{D}[C', u] \mid C' \subset C, |C'| = |C| - 1\}$. We will exploit this result also later. In conclusion, the overall running time of the TraversePowerset-BruteForce algorithm is $\mathcal{O}(2^{|L|}k(m + n|L|))$.

3.2.2 Pruning the search space

We now define a number of pruning rules to improve the efficiency of the TraversePowerset-BruteForce algorithm. We focus our discussion on a single landmark x. Our pruning rules are classified into three categories:

- *Skipping unnecessary label sets*, i.e., recognize early the label sets C for which there exists no vertex u such that C is SP-minimal with respect to x and u.
- *Skipping unnecessary* SP*-minimality tests*, i.e., once a C-constrained SSSP with source x has been computed, identify a set of vertices for which C cannot be SP-minimal and skip the corresponding SP-minimality test.
- *Speeding-up* SP*-minimality tests*, i.e., for some vertices u, recognize if a label set C is SP-minimal or not with respect to x and u more efficiently than $\mathcal{O}(|C|)$ time.

We discuss next the three categories in more detail.

Skipping unnecessary label sets. For any given landmark x, the labelsets that can safely be discarded are those label sets C for which the set of vertices reachable from x is empty, i.e., those label sets C such that $d_C(x,u) = \infty$, for all $u \in V$. Early detection of such label sets can be carried out based on the observation that a label set C yields an empty set of vertices reachable from x *if and only if* C contains no labels present on the edges incident to x. In that case (and only in that case), there is no way for the landmark x to remain connected to the rest of the graph. This observation is formalized next.

OBSERVATION 1. *Given an edge-labeled graph $G = (V, E, L, \ell)$ and a landmark $x \in X$, let L_x be the set of all labels placed on edges incident to x, i.e., $L_x = \{\ell(x,u) \mid (x,u) \in E\}$. For any label set $C \subseteq L$ it holds that: $d_C(x,u) = \infty$ for all $u \in V$ if and only if $C \subseteq L \setminus L_x$.* □

To exploit Observation 1, we modify the algorithm TraversePowerset-BruteForce as follows. Instead of visiting *all* $C \subseteq L$ (Lines 4-6 in Algorithm 1), we avoid generating unnecessary label sets by employing a strategy that resembles candidate generation of the well-known Apriori algorithm for frequent-itemset mining [2].

While Apriori is a level-wise bottom-up strategy, in our setting, we need to generate candidates (i.e., label sets) in a top-down fashion. However, with a simple trick, we can still rely on a bottom-up strategy and keep all its efficiency advantages. The idea is to generate candidates in a standard bottom-up fashion, while testing the pruning condition in Observation 1 on the *complement* of each candidate C. More precisely, given a candidate C, we decide whether it should be filtered out by checking if $L \setminus C$ is a subset of $L \setminus L_x$ (where $L_x = \{\ell(x,u) \mid (x,u) \in E\}$), or, equivalently, whether $C \supseteq L_x$. The details of the procedure just described are reported as Function 1.

Function 1 GenerateCandidates(G, x)

Input: an edge-labeled graph $G = (V, E, L, \ell)$, a landmark x
Output: a set of label sets $\mathcal{C}$
1: $\mathcal{C} \leftarrow \emptyset$
2: $L_x \leftarrow \{\ell(x,u) \mid (x,u) \in E\}$
3: Cand $\leftarrow \{\{l\} \mid l \in L\}$
4: **while** Cand $\neq \emptyset$ **do**
5: Cand $\leftarrow$ Cand $\setminus \{C \mid C \in \text{Cand}, C \supseteq L_x\}$
6: $\mathcal{C} \leftarrow \mathcal{C} \cup \{(L \setminus C) \mid C \in \text{Cand}\}$
7: Cand $\leftarrow$ AprioriNextLevel(Cand)
8: **end while**
9: **return** $\mathcal{C}$

Skipping unnecessary SP**-minimality tests.** Once a C-constrained SSSP with source x has been computed for any label set C, the SP-minimality of C should in principle be checked with respect to all vertices u having distance $d_C(x,u) < \infty$ (Lines 7-11 in Algorithm 1). Here we show how to early recognize vertices u for which C cannot be SP-minimal, thus avoiding to take into consideration such vertices at all. Particularly, we observe the following: for any label set C to be SP-minimal with respect to a vertex u (and a landmark x) each label in C must be present on at least one edge of every C-constrained shortest path between x and u; otherwise, C cannot be SP-minimal with respect to u because there would exist a shortest path between x and u that uses only a subset of C. An immediate consequence of this is that C cannot be SP-minimal with respect to any vertex u having distance $d_C(x,u) < |C|$. We formalize this observation next.

OBSERVATION 2. *Given a landmark $x \in X$ and a vertex $u \in V$, any label set $C \subseteq L$ is* SP*-minimal with respect to x and u only if $d_C(x,u) \geq |C|$.* □

To profitably exploit the above observation, the output of any C-constrained SSSP with source x can be organized into an appropriate data structure where, given a distance t, all objects u having $d_C(x,u) = t$ can be accessed in constant time. This way, we can process only vertices at a distance $t \geq |C|$ and skip all other ones.

Speeding-up SP**-minimality tests.** Finally, we discuss two observations that allow the SP-minimality of certain label sets C and vertices u be checked in $\mathcal{O}(1)$ time rather than $\mathcal{O}(|C|)$.

The first observation is the following. Consider a vertex u and all *unconstrained* shortest paths connecting u to the landmark x. Assume that at least one of those shortest paths is monochromatic. Denote by l_u the unique label of that path. Then, it is easy to see that any label set C containing l_u cannot be SP-minimal with respect to x and u. We formalize this observation next.

OBSERVATION 3. *Given a landmark $x \in X$, let u be a vertex in V such that there exists an unconstrained shortest path between x and u that is monochromatic and let l_u denote the unique label on the edges of such a monochromatic path. It holds that all label sets $C \supset \{l_u\}$ are non-*SP*-minimal with respect to x and u.* □

To exploit Observation 3 in practice, we compute and store the set V_x of all vertices u that have a monochromatic shortest path connecting them to x, along with the unique label l_u of the corresponding path for every u. This can be achieved in $\mathcal{O}(m + n)$

Algorithm 2 TraversePowerset

Input: an edge-labeled graph $G{=}(V, E, L, \ell)$, a set of landmarks X
Output: for each pair (x, u), where $x \in X$ and $u \in V$, a set $\mathcal{SP}_{xu}$ of $\langle C, d\rangle$ pairs storing all SP-minimal label sets C with respect to x and u along with the corresponding C-constrained shortest path distance d

1: $\mathcal{SP}_{xu} \leftarrow \emptyset, \forall x \in X, u \in V$
2: **for all** $x \in X$ **do**
3: $\quad\mathbf{D} \leftarrow \emptyset, \mathbf{V} \leftarrow \emptyset$
4: $\quad\mathcal{C} \leftarrow$ GenerateCandidates(G, x) {Observ. 1}
5: $\quad$**for all** $C \in \mathcal{C}$ **do**
6: $\quad\quad\langle \mathbf{D}[C], \mathbf{V}[C]\rangle \leftarrow$ ConstrainedSSSP(G, x, C)
7: $\quad$**end for**
8: $\quad\mathbf{L} \leftarrow$ SingleLabelSP$(G, x, \mathbf{D}[L], \mathbf{V}[L])$ {Observ. 3}
9: $\quad$**for all** $C \in \mathcal{C}$ **do**
10: $\quad\quad$**for all** $t \geq |C|$ {Observ. 2} **do**
11: $\quad\quad\quad\widehat{V}_t \leftarrow$ nonSPminimal$(C, \mathbf{V}[C, t-1], \mathbf{V}[C, t])$ {Observ. 4}
12: $\quad\quad\quad$**for all** $u \in \widehat{V}_t, \{\mathbf{L}[u]\} \not\subset C$ {Observ. 3} **do**
13: $\quad\quad\quad\quad$**if** C is SP-minimal w.r.t. x and u **then**
14: $\quad\quad\quad\quad\quad\widehat{V}_t \leftarrow \widehat{V}_t \setminus \{u\}$
15: $\quad\quad\quad\quad$**end if**
16: $\quad\quad\quad$**end for**
17: $\quad\quad\quad\mathcal{SP}_{xu} \leftarrow \mathcal{SP}_{xu} \cup \{\langle C, \mathbf{D}[C, u]\rangle\}, \forall u \in \mathbf{V}[C, t] \setminus \widehat{V}_t$
18: $\quad\quad$**end for**
19: $\quad$**end for**
20: **end for**

time by computing an unconstrained SSSP with source x and then visiting the output of such a SSSP level-by-level, i.e., starting from vertices at distance 1 from x and proceeding by increasing the distance one-by-one. This information can profitably be exploited every time an SP-minimality check is required for any label set C and vertex $u \in V_x$: C can be recognized as non-SP-minimal with respect to x and u in constant time by simply checking whether the label l_u belongs to C.

Let us now discuss our second observation. We recall that, to exploit Observation 2, the output of any C-constrained SSSP is organized as a collection of vertex sets accessible in constant time based on the distance from the landmark x. Let V_t denote the vertex set at distance t from x. The SP-minimality of C is checked distance-by-distance, starting from the set $V_{|C|}$. Then, for any vertex within V_t, we can safely assume that the SP-minimality of all vertices in V_{t-1} has already been checked. We can exploit this as follows. Given a vertex $u \in V_t$, let $V_{t-1,u}$ denote the set of all vertices in V_{t-1} connected to u by an edge whose label belongs to the constraint label set C, i.e., $V_{t-1,u} = \{v \in V_{t-1} \mid (u, v) \in E, \ell(u, v) \in C\}$. Our observation is: a label set C is SP-minimal with respect to a vertex $u \in V_t$ and a landmark x if C has been recognized as SP-minimal with respect to *every vertex* in $V_{t-1,u}$. Indeed, it is easy to see that, if such a condition holds, then there cannot exist any shortest path connecting u to x whose labels are a subset of C. Hence, C is SP-minimal with respect to u as well.

OBSERVATION 4. *Given a landmark $x \in X$ and a label set $C \subseteq L$, let V_t denote the set of all vertices u having distance $d_C(x, u) = t$ and $V_{t-1,u} = \{v \in V_{t-1} \mid (u, v) \in E, \ell(u, v) \in C\}$. For any vertex $u \in V_t$, if C is* SP*-minimal with respect to all vertices $v \in V_{t-1,u}$ (and x), then C is* SP*-minimal with respect to u (and x).* □

According to the above observation, the SP-minimality check of C can be limited only to vertices $v \in V_t$ having at least one non-SP-minimal neighbor in V_{t-1}. The data structures to find such vertices are easily produced by the C-constrained SSSP algorithm. For all other vertices in V_t we can conclude that C is SP-minimal without further computations.

3.2.3 *The* TraversePowerset *algorithm*

The algorithm that exploits the pruning rules described in Section 3.2.2 is dubbed as TraversePowerset (Algorithm 2).

First, Observation 1 (and Function 1) is exploited to avoid generating unnecessary label sets (Line 4), and, hence, compute label-constrained SSSPs only for the necessary label sets (Lines 5-7). Note that a C-constrained SSSP returns the distance of each vertex $u \in V$ from x (vector $\mathbf{D}[C]$), as well as a vector of vertex sets indexed by the distance from x (vector $\mathbf{V}[C]$). We also assume that all edges with label in C connecting vertices between two consecutive vertex sets $\mathbf{V}[C, t-1]$, $\mathbf{V}[C, t]$ are available. To exploit Observation 3, a vector $\mathbf{L}$ is computed by function SingleLabelSP (Line 8): $\mathbf{L}$ contains, for all vertices u, the label of the monochromatic shortest path between u and x (if any).

In the main cycle of the algorithm (Lines 9-19), all label sets within $\mathcal{C}$ are processed. Based on Observation 2, the vertices taken into account for each $C \in \mathcal{C}$ are only those at distance $\geq |C|$ from x (Line 10). Among such vertices, the subset of those ones for which C cannot immediately be recognized as SP-minimal based on Observation 4 is firstly identified (Line 11). Finally, only the vertices in the latter subset for which Observation 3 does not apply (Line 12) enter a "standard" SP-minimality test, i.e., an SP-minimality test performed based on Theorem 2 (Lines 13-15).

3.3 Selecting landmarks

Exact landmark selection. We now turn our attention to the problem of selecting good landmarks for the PowCov index. We first formalize the POWCOV-LANDMARK-SELECTION problem that asks to find a minimum-sized landmark set that allows the PowCov index to answer *all* queries *exactly*.

DEFINITION 3. *Given a set of landmarks X and a query $Q = \langle s, t, C\rangle$, let $\tilde{d}_{PC}(Q, X)$ denote the approximate answer to Q provided by the PowCov index using the landmarks X. A set of landmarks X is called PowCov-exact if and only if $\tilde{d}_{PC}(Q, X) = d_C(s, t)$, for all queries $Q = \langle s, t, C\rangle$.* □

PROBLEM 1 (POWCOV-LANDMARK-SELECTION). *Given an edge-labeled graph $G = (V, E, L, \ell)$, find a minimum-sized set of landmarks $X \subseteq V$ such that X is PowCov-exact.* □

A first step for tackling Problem 1 is to determine the conditions under which any landmark set X is PowCov-exact. This is stated in the following lemma.

LEMMA 1. *Given an edge-labeled graph $G = (V, E, L, \ell)$, a set of landmarks $X \subseteq V$ is PowCov-exact if and only if, for all pairs of vertices u, v and for all* SP*-minimal label sets C with respect to u and v, there exists a landmark in X lying on at least one C-constrained shortest path $p^*_C(u, v)$.*

PROOF. Let us prove the first direction of the logical implication. First, by the notion of SP-minimality, the condition that there exists a C-constrained shortest path that intersects X for all vertices u, v and for all SP-minimal label sets C with respect to u and v actually holds for *all* label sets, not only the SP-minimal ones. This means that, for each query $\langle s, t, C\rangle$, there exists a landmark lying on at least one C-constrained shortest path between s and t. This is sufficient for the upper bound given by triangle inequality to be exact, provided that the intermediate distances $d_C(s, x)$ and $d_C(x, t)$ stored in the index considered are exact for each input query. The latter condition is ensured by PowCov, therefore the claim follows.

The other side of the implication states that if there exists a pair u, v and an SP-minimal label set C (with respect to u and v) such

that all C-constrained shortest paths $p^*_C(u,v)$ do not pass through any landmark in X, then X is not PowCov-exact. If this arises, then a query $\langle u,v,C\rangle$ is answered by PowCov resorting to the upper bound computed according to a path $p^*_S(u,v)$ that involves some subset $S \subset C$. The SP-minimality of C guarantees that $d_S(u,v) > d_C(u,v)$, then the query $\langle u,v,C\rangle$ is not answered exactly. □

Based on Lemma 1, we can now derive a key result that relates the POWCOV-LANDMARK-SELECTION problem with the *vertex-cover* problem. Recall that a vertex cover of a graph $G=(V,E)$ is a subset of vertices $V' \subseteq V$ such that for all edges $(u,v) \in E$ it is either $u \in V'$ or $v \in V'$. The relation is stated in the next theorem.

THEOREM 3. *Given an edge-labeled graph $G=(V,E,L,\ell)$, a set of landmarks $X \subseteq V$ is PowCov-exact if and only if X is a vertex cover of G.*

PROOF. We prove the theorem by showing that any landmark set X satisfies Lemma 1 if and only if it is a vertex cover of G. First, by definition of vertex cover, it is guaranteed that, for each pair of vertices u,v in G and for each path p between u and v, there exists at least one landmark $x \in X$ belonging to p. This implies that, for each query label set C, at least one landmark lies on a C-constrained shortest path between u and v. Thus, X satisfies Lemma 1.

On the other hand, note that if X is not a vertex cover of G, then there exists an edge (u,v) in G such that $u \notin X$ and $v \notin X$. This means that the query $\langle u,v,C\rangle$, where $C=\{\ell(u,v)\}$, cannot be answered exactly. Indeed, $d_C(u,v)$ is clearly equal to 1, but, as neither u nor v belongs to the landmark set X, the bound provided for this query is necessarily ≥ 2 because it is computed on a path passing through at least one vertex other than u and v. It easy to see that the label set $C=\{\ell(u,v)\}$ is SP-minimal with respect to u and v and the path composed by the single edge (u,v) is the only C-constrained *shortest* path between u and v. Thus, we have found a pair of vertices u,v and an SP-minimal label set with respect u and v such that no C-constrained shortest path between u and v contains a landmark in X. This makes X violating the condition required by Lemma 1. The theorem follows. □

An immediate corollary of Theorem 3 is that a minimum vertex cover is a solution for Problem 1.

COROLLARY 1. *A set of landmarks X is a solution for the* POWCOV-LANDMARK-SELECTION *problem if and only if X is a minimum vertex cover of the input graph.* □

Approximate landmark selection. Corollary 1 suggests to select landmarks by finding a minimum vertex cover of the graph G. However, even though the size of the minimum vertex cover may be efficiently approximated within a factor 2 [15], this size, in many cases in practice, may be $\Omega(n)$ This is too large for our index, as the basic assumption for any landmark-based index is is to have a number of landmarks $\ll n$.

To overcome this drawback, we depart from the requirement of finding a set of landmarks that allows answering *all* queries exactly. Instead, we aim at *maximizing the number of queries that can be answered exactly* with k landmarks. However, considering a problem formulation where the optimal k landmarks explicitly maximize the number of queries correctly answered may lead to inefficient optimization strategies. The intuition behind this is that, while in the non-labeled case assessing the number of queries exactly answered by a single landmark needs one SSSP and time $\mathcal{O}(m)$, the labeled case requires $2^{|L|}$ SSSPs (one for each query label set) and time $\mathcal{O}(m2^{|L|})$, which makes even simple heuristics like local search not scalable. For this reason, and further motivated by the above results about vertex covering, we formulate the problem as a k-MAX-VERTEX-COVER problem [7]: given an integer k, find k vertices in the input graph so that the number of covered edges is maximized. Note that such a formulation is still close to the formulation that explicitly considers the number of queries exactly answered, as the more the edges covered by a landmark set, the more the queries answered correctly. At the same time, this formulation has the advantage that can be (approximately) solved efficiently, as shown next.

The k-MAX-VERTEX-COVER problem is **NP**-hard [7], but it admits a simple and efficient approximation algorithm which we call GreedyMVC. Given the input graph and a partial solution containing $< k$ landmarks (initially empty), the algorithm always selects the vertex that covers the largest number of still uncovered vertices. That is, the vertex that maximizes the marginal gain of the cover is added to the set of landmarks in each iteration, until k landmarks have been collected. Using standard arguments for submodular-function maximization [15], the GreedyMVC algorithm can be shown to achieve an approximation factor of $\left(1-\frac{1}{e}\right) \approx 0.632$. This means that the solution we obtain for k landmarks covers at least 63% of the vertices covered by the optimal solution with k landmarks. By exploiting this and adapting another result stated in [7], we obtain the following theorem:

THEOREM 4. *The GreedyMVC algorithm used for selecting landmarks for the PowCov index provides an approximation guarantee of* $\max\left\{1-\frac{1}{e},\frac{k}{n}\right\}$.

PROOF. Given a graph $G=(V,E)$, let $VC=[u_1,\ldots,u_k] \subset V$ be the output of GreedyMVC, ordered based on the specific iteration where vertices are added to VC. Let also δ_i denote the number of edges covered by u_i in the graph resulting after the executions of iterations 1 through $i-1$, $\forall i \in [1..k]$. Finally, let VC^* denote the optimal solution, and $E[VC]$ and $E[VC^*]$ the total number of edges covered by VC and VC^*, respectively.

First, we note that $\sum_{i=1}^{k}\delta_i = E[VC]$, while $\sum_{i=1}^{n}\delta_i = |E| \geq E[VC^*]$. Also, as in each iteration GreedyMVC picks the maximum degree vertex, where the degree is computed based on the reduced graph resulting from all previous iterations, it holds that $\delta_1 \geq \delta_2 \geq \cdots \geq \delta_k$. This implies that $\sum_{i=1}^{k}\delta_i \geq \frac{k}{n}\sum_{i=1}^{n}\delta_i$. Combining all these findings, it results that $E[VC] = \sum_{i=1}^{k}\delta_i \geq \frac{k}{n}\sum_{i=1}^{n}\delta_i = \frac{k}{n}E[VC^*]$. Thus, GreedyMVC is a $\frac{k}{n}$-approximation algorithm for k-MAX-VERTEX-COVER.

Combining this result with the result derived from submdular-function maximization we obtain an overall approximation factor of $\max\left\{1-\frac{1}{e},\frac{k}{n}\right\}$. □

4. CHROMATIC LANDMARKS INDEX

In the worst case, the construction time of the PowCov index remains exponential in the number of labels. This is because fundamentally PowCov is based on the same strategy as the brute-force approach, it only uses a number of efficient pruning heuristics.

In this section, we propose a second index, called Chromatic Landmarks (ChromLand), which is kept as light as possible and similar to the standard landmark approach for non-labeled graphs. As a trade-off, ChromLand incurs in increased query time and offers less accurate answers.

4.1 Structure of the index

The main idea of the ChromLand index is that each of the k landmarks is *assigned* to a single label (color) in L. The land-

marks of this index are called *chromatic landmarks*. Any distance involving one or more of such landmarks is referred to as *chromatic distance*. In particular, the chromatic distance between a landmark $x \in X$ and a vertex $u \in V$ is defined as the distance $d_{\{c(x)\}}(x,u)$ computed using only edges having the color $c(x)$ assigned to x, while the chromatic distance between any two landmarks $x, y \in X$ corresponds to the distance $d_{\{c(x),c(y)\}}(x,y)$ using only edges of the colors of both x and y. We denote by $cd(\cdot,\cdot)$ the chromatic distance, and thus, we have $cd(x,u) = d_{\{c(x)\}}(x,u)$—vertex-to-landmark; or $cd(x,y) = d_{\{c(x),c(y)\}}(x,y)$—landmark-to-landmark.

The structure of the ChromLand index is thus simple: for each vertex $u \in V \setminus X$ we store the (mono-)chromatic distances to all landmarks, and for each landmark $x \in X$ we store the (bi-)chromatic distances to all other landmarks. Building and storing the index can be accomplished with k BFS traversals, requiring $\mathcal{O}(km)$ time and $\mathcal{O}(kn)$ space.

4.2 Query processing

A simple way to process LC-PPSPD queries using the ChromLand index, is to use the upper-bound based on the triangle inequality, as discussed in the Introduction.

PROPOSITION 2. *Given a query $\langle s,t,C\rangle$ and a set of chromatic landmarks X, it holds that*

$$d_C(s,t) \leq \min\{cd(x,s) + cd(x,t) \mid x \in X \text{ and } c(x) \in C\}.$$

PROOF. First, we observe that $d_C(s,t) \leq d_S(s,t)$, $\forall S \subseteq C$. Combining this with triangle inequality leads to:

$$\begin{aligned} d_C(s,t) &\leq d_C(x,s) + d_C(x,t),\ \forall x \in X \\ &\leq d_{\{c(x)\}}(x,s) + d_{\{c(x)\}}(x,t),\ \forall x \in X \text{ s.t. } c(x) \in C \\ &= cd(x,s) + cd(x,t),\ \forall x \in X \text{ s.t. } c(x) \in C, \end{aligned}$$

which clearly implies that $d_C(s,t) \leq \min\{(cd(x,u) + cd(x,t)) \mid x \in X \wedge c(x) \in C\}$. □

This query-processing strategy requires $\mathcal{O}(k)$ time, like the landmark approach for standard PPSPD queries. Nevertheless, the above approach may result in poor accuracy as distances in the precomputed index consider only monochromatic paths, and the shortest-path distance for larger constraint label sets may not be accurately approximated by a monochromatic path.

We handle the above issue by considering paths that pass through more landmarks. We remark that, for PPSPD queries on non-labeled graphs, using multiple landmarks does not yield any improvement, because the length of a path using landmark x is upper bounded by the length of a path using landmarks x and y. Instead, in the multi-chromatic context, the estimation can be improved by using more landmarks. Indeed, if two chromatic landmarks x and y are assigned to different colors, the following upper bound for $d_C(u,v)$ could be tighter than the simpler one in Proposition 2 (see Figure 3 for an example):

$$\begin{aligned} d_C(s,t) \leq \min\{&cd(u,x) + cd(x,y) + cd(y,v) \mid \\ &x,y \in X \text{ and } c(x), c(y) \in C \text{ and } c(x) \neq c(y)\}. \end{aligned}$$

The above observation can be generalized to a sequence of landmarks $x_1, \ldots, x_z \in X$, where any two consecutive landmarks have different colors. Next we show how to use multiple landmarks in order to get even tighter bounds for a given LC-PPSPD query. Recall that the ChromLand index is essentially a table storing mono-chromatic distances between landmark-vertex pairs and bi-chromatic distances between landmark-landmark pairs. Intuitively, one can think of this index as an *auxiliary* graph $G_X = (V, X, E_X, c, w)$, where V is the set of vertices of the original graph G, $X \subseteq V$ is the set of landmarks, and $c : X \to L$ is a function that assigns landmarks to colors. The edge set E_X of this auxiliary graph is defined as follows: there exists an edge between any landmark-vertex pair (x,u) if and only if $cd(u,x) < \infty$; there exists an edge between any landmark-landmark pair (x,y) if and only if $c(x) \neq c(y)$ and $cd(x,y) < \infty$. Each edge in E_X is labeled with the color(s) of the incident landmark(s) and is weighted by a function $w : E_X \to \mathbb{N}$ defined as $w(u,v) = cd(u,v)$. We obtain the desired bound by exploiting the following result.

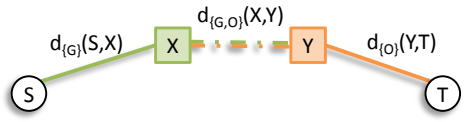

Figure 3: ChromLand query-processing strategy for the query $\langle s,t,\{\text{green,orange}\}\rangle$. The distance $d_{\{\mathbf{g},\mathbf{o}\}}(s,t)$ is approximated by a path passing trough two landmarks, x and y. The shortest path from s to t might not pass through x or y, but its length is upper bounded by $d_{\{\mathbf{g}\}}(s,x) + d_{\{\mathbf{g},\mathbf{o}\}}(x,y) + d_{\{\mathbf{o}\}}(y,t)$. This improves upon using only x if $d_{\mathbf{g}}(t,x) > d_{\{\mathbf{g},\mathbf{o}\}}(x,y) + d_{\{\mathbf{o}\}}(y,t)$.

THEOREM 5. *Let $G = (V,E,L,\ell)$ be an edge-labeled graph and $X \subseteq V$ a set of landmarks. Let $G_X = (V,X,E_X,c,w)$ be the auxiliary graph of G defined over G and X. Given a label set C and two vertices $u, v \in V$, let $G_X[u,v,C]$ denote the subgraph of G_X induced by the set of vertices $\{u,v\} \cup \{x \in X \mid c(x) \in C\}$. For any query $\langle s,t,C\rangle$ the shortest path distance $\delta_C(s,t)$ between s and t computed on $G_X[s,t,C]$ is the tightest upper bound to $d_C(s,t)$ that can be computed from the information stored by ChromLand index.*

PROOF. Given any $Y \subseteq X$, let $sp_Y(s,t)$ be the shortest-path distance computed over the subgraph of G_X induced by the vertices in $\{s,t\} \cup Y$. It holds that $c(y) \in C, \forall y \in Y \Rightarrow d_C(s,t) \leq sp_Y(s,t)$, as $d_C(u,v) \leq cd(u,v)$, $\forall u,v \in \{s,t\} \cup Y$. Conversely, it results that $\exists y \in Y$ such that $c(y) \notin C \not\Rightarrow d_C(s,t) \leq sp_Y(s,t)$, because, this way, there might exist a pair of vertices $u,v \in \{s,t\} \cup Y$ such that $d_C(u,v) > cd(u,v)$, and this could violate the overall upper bound $d_C(s,t) \leq sp_Y(s,t)$. For this purpose, only subsets $Y \subseteq X$ whose landmarks are all coupled with colors within the query label set C guarantee sound upper bounds to the distance $d_C(s,t)$. The tightest among these sound bounds is defined by taking the largest of these sets Y, i.e., $Y = \{x \mid x \in X, c(x) \in C\}$. This means that the tightest, sound upper bound for $d_C(s,t)$ given the information in ChromLand is equal to the shortest-path distance computed on the subgraph of G_X induced by the vertices $\{s,t\} \cup \{x \mid x \in X, c(x) \in C\}$, i.e., $G_X[s,t,C]$. □

The theorem suggests that to approximate $d_C(s,t)$ for a query $\langle s,t,C\rangle$ we need to consider the subgraph of G_X induced by the vertices s, t, and all landmarks in X whose color belongs to C, and then compute the (exact) shortest-path distance between s and t on that subgraph. This strategy requires running a shortest-path algorithm (e.g., Dijkstra) on a weighted graph with $\mathcal{O}(k)$ vertices and $\mathcal{O}(k^2)$ edges; thus, the query processing of the enhanced ChromLand index requires $\mathcal{O}(k^2)$ time. Note that since $k^2 \ll n$, this strategy is faster than computing exact distances without any index. An illustration of the entire process is reported in Figure 4.

4.3 Selecting landmarks

A key result in selecting landmarks for the PowCov index, stated in Lemma 1, is that the index can provide the exact answer to a

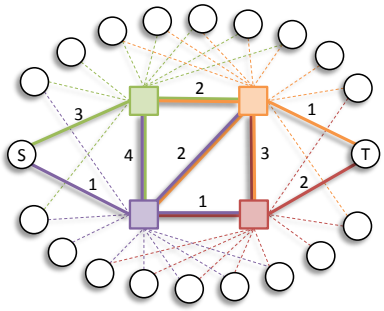

Figure 4: Illustration of a ChromLand index. Each landmark (square) is assigned to a color. Each edge is labeled with the color(s) of the landmark(s) and weighted by the corresponding chromatic distance $cd(\cdot,\cdot)$. Given the query $\langle s, t, \{\text{green, orange, red}\}\rangle$, we take the subgraph induced by s and t and all landmarks whose color is in $\{\text{g, o, r}\}$, and we approximate $d_{\{\text{g,o,r}\}}(s,t)$ by computing the shortest-path distance between s and t on that subgraph. In this example the shortest path is passing only trough the green and orange landmarks, and has length 6. Note that all bounds defined by a single landmark give infinite distances.

query $\langle s, t, C\rangle$ only if there exists at least one landmark on a shortest path $p^*_C(u,v)$. This statement does not hold for the ChromLand index. Indeed, the following theorem shows that a single landmark on $p^*_C(u,v)$ does not suffice; instead, the number of landmarks should be at least as large as the number of distinct colors on $p^*_C(u,v)$.

THEOREM 6. *Given an edge-labeled graph $G = (V, E, L, \ell)$, a set of landmarks $X \subseteq V$ allows the ChromLand index to provide exact answers only if for all pairs $u, v \in V$ and all label sets $C \subseteq L$, there exists a shortest path $p^*_C(u,v)$ such that $|X \cap \{i \mid (i,j) \in p^*_C(u,v)\}| \geq |\text{colors}(p^*_C(u,v))|$.*

PROOF. Given any two vertices $u, v \in V$ and a label set $C \subseteq L$, let $\mathcal{P}^*_C(u,v)$ be the set of all C-constrained shortest path between u and v. To prove the claim, it is sufficient to show that, if X does not contain at least $h = |colors(p^*_C(u,v))|$ landmarks lying on some $p^*_C(u,v) \in \mathcal{P}^*_C(u,v)$, then there exists at least one query that ChromLand cannot answer exactly when employing X. To this end, note that, as each landmark has assigned only one color, if the landmarks lying on $p^*_C(u,v)$ are less than h, for all $p^*_C(u,v) \in \mathcal{P}^*_C(u,v)$, then there would be at least one color not "covered" by any landmark on each $p^*_C(u,v)$. This means that the ChromLand index actually considers at least one edge in each $p^*_C(u,v)$ as missing. Thus, whatever colors are assigned to the landmarks, the answer provided by ChromLand to a query $\langle u, v, C\rangle$ would be always $> |p^*_C(u,v)|$, as it would be computed by considering a path not belonging to $\mathcal{P}^*_C(u,v)$, and, therefore, not shortest. □

The theorem suggests that landmark selection in ChromLand is more complex than in PowCov. For instance, a vertex cover represents no longer a valid solution. As an example, consider the simple graph G shown in Figure 5. Although the set $X = \{x\}$ is a vertex cover of G, it is not a valid landmark set for ChromLand—whatever color is assigned to x, there is no way to provide exact answers to queries involving label sets with size larger than 1.

Problem formulation. Consider again the example shown in Figure 3: the shortest-path distance $d_{\{\text{g,o}\}}(s,t)$ is approximated by the sum of three chromatic distances, i.e., $d_{\{\text{g}\}}(s,x) + d_{\{\text{g,o}\}}(x,y) + d_{\{\text{o}\}}(y,t)$. Note that the smaller these chromatic distances are, the tighter is the approximation going to be. Motivated by this example, we focus on selecting a set of landmarks so that *any vertex of the graph is close to at least one landmark for any given color*.

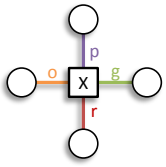

Figure 5: Simple edge-labeled graph G where the vertex cover $\{x\}$ is not a valid landmark set for ChromLand.

Algorithm 2 ChromLandLocalSearch

Input: an edge-labeled graph $G = (V, E, L, \ell)$, an integer k
Output: a set X of k landmarks, a landmark labeling function c

```
⟨X, c⟩ ← randomSelect(V, L, k)
J* ← J(G, X, c)
repeat
    randomly pick: a vertex u ∈ V \ X, a landmark x ∈ X, a color l ∈ L
    ⟨X', c'⟩ ← swap(X, c, u, x, l)
    if J(G, X', c') > J* then
        ⟨X, c⟩ ← ⟨X', c'⟩, J* ← J(G, X', c')
    end if
until stop
```

We translate this intuition to an optimization problem. First, given a landmark $x \in X$, a vertex $u \in V$, and a landmark-labeling function $c : X \to L$, we define the similarity function sim_c as follows:

$$\text{sim}_c(x,u) = \begin{cases} 0 & \text{if } d_{\{c(x)\}}(x,u) = \infty, \\ d_{\{c(x)\}}(x,u)^{-1} & \text{otherwise.} \end{cases}$$

We then define our landmark-selection problem as follows.

PROBLEM 2 (CHROMLAND-LANDMARK-SELECTION). *Given an edge-labeled graph $G = (V, E, L, \ell)$ and an integer k, find a set of k landmarks $X \subseteq V$ and a landmark-labeling function $c : X \to L$ so as to maximize the objective function*

$$J(G, X, c) = \sum_{u \in V} \max_{x \in X} \text{sim}_c(x,u). \quad \square$$

A solution based on k-MEDIAN. The CHROMLAND-LANDMARK-SELECTION problem can be interpreted as a variant of the k-MEDIAN problem [3]. Specifically, we map CHROMLAND-LANDMARK-SELECTION to a variant of k-MEDIAN as follows: let (M, D, H) be a bipartite graph, where M and D are disjoint sets of vertices, and H is the set of edges. The set D, representing "demand" points, is a copy of the vertex set V in the original graph. The set M, representing "median" points, is a copy of all vertex-color pairs $V \times L$. The weight of an edge $(\langle x, c_x\rangle, u) \in M \times D$ is $\text{sim}_c(x,u)$.

The goal of the standard k-MEDIAN problem is to select a set of median points $M^* \subseteq M$ so that all demand points in D are served by their closest median point in M^* and the total service cost is minimized. Here, by defining the edge weights using the similarity function sim_c we cast the problem as a maximization problem.

Note that setting the set of medians M equal to $V \times L$ allows us to also determine the landmark-labeling function: if a median point $\langle x, c_x\rangle$ is selected in the solution set M^*, we then set the color of x equal to c_x. To make this work, we need to impose the additional constraint that the set M^* contains only distinct landmarks, i.e., for all distinct pairs $\langle x, c_x\rangle, \langle y, c_y\rangle \in M^*$ we require that $x \neq y$.

A key challenge is to design an algorithm that does not compute/materialize all pairwise similarities, as this would require unaffordable $\Omega(n^2)$ time/space. The solution we propose is an adap-

Table 1: Characteristics of the selected datasets.

dataset	*# vertices*	*# edges*	*# labels*	*diameter*	*# queries*
BioGrid	26 806	298 957	7	18	19 037
BioMine	943 510	5 727 448	7	16	20 799
String	1 490 098	8 886 639	6	19	18 149
DBLP	47 598	252 881	8	19	18 611
Youtube	15 088	19 923 067	5	6	23 499
synthetic	500 000	2 500 000	4–100	$[5, 20]$	$\sim$ [15K, 100K]

tation of the *local-search* heuristic for k-MEDIAN [3] and is outlined as Algorithm 2. that works as follows. The algorithm starts with a randomly chosen set of medians M^*. It then tries to swap one of the current medians with another point that is is not currently in M^*. If the swap improves the current objective-function score, then the swap is performed. The process is repeated for a fixed number of iterations. As a result, for a number of iterations equal to I, the time complexity of the algorithm is $\mathcal{O}((I + k)m)$. The space complexity is instead $\mathcal{O}(nk)$, as it requires keeping the distance of each point to its closest median.

5. EXPERIMENTAL EVALUATION

We experiment with both real and synthetic data, whose main characteristics are shown in Table 1. Synthetic datasets are generated using the generator described in [6]. As far as real datasets, BioGrid and String are protein-interaction networks obtained from `thebiogrid.org` and `string-db.org`. The two datasets are undirected graphs, whose vertices correspond to proteins and each edge is labeled with the type of the interaction occurring between the two adjacent vertices (proteins). BioMine is a recent snapshot of the database of the BioMinE project (`biomine.brgm.fr`), which is a collection of biological interactions. Each interaction among vertices of the graph has a label denoting the type of the interaction. DBLP is a subset of the popular co-authorship network taken from [6]. The dataset is modeled in [6] as an edge-labeled graph where an edge exists among any two authors if they have co-authored at least once. The label on each edge is derived by considering the bag of words obtained by merging the titles of all papers coauthored by the two authors, and applying *Latent Dirichlet Allocation* (LDA) [4] to automatically identify a topic distribution on each edge. The edge label is ultimately assigned by taking the most likely topic discovered for that edge. Youtube is a subset of the users of the popular video-sharing service used in [26] and publicly available at `socialcomputing.asu.edu/datasets/YouTube`. The edges of such a dataset have been labeled by the authors of [26] by using one of the following user relationships: *contact*, *co-contact*, *co-subscription*, *co-subscribed*, and *favorite*.

The query workload of each dataset is obtained by first sampling 5 000 random pairs of connected vertices. Then, for each pair, we pick $|L|$ random label sets of sizes $1, 2, \ldots, |L|$. This results in $5\,000 \times |L|$ distinct queries for each dataset. Out of these, we keep only the ones corresponding to finite distances: there is no need to consider unreachable pairs as the proposed indexes guarantee that no false pasitives can arise. The exact number of queries considered for each dataset is in Table 1 (last column).

We implemented our code in JAVA and run experiments on a single core of an Intel Xeon server at 2.83GHz CPU and 64GB RAM.

Unless otherwise specified, the proposed PowCov and ChromLand indexes are always equipped with landmarks selected according to the strategies presented in Sections 3.3 and 4.3, respectively.

5.1 Index construction

We focus here on the index building phase, by relying on real and synthetic datasets. In particular, a major purpose of involving synthetic datasets is to analyze the behavior with varying the labels.

Index size. Table 2 summarizes the index size on the selected datasets, reported as average number of distances stored per landmark-vertex pair (unreachable vertices are omitted from the counting). We report results for our PowCov, as well as for the index resulting from the naïve adaptation of the landmark paradigm to the context of edge-labeled graphs discussed in the Introduction, i.e., the index that stores a number of per-landmark-vertex-pair distances exponential in the number of labels. Particularly, the size of any single landmark-vertex pair reported by the naïve index refers to all label sets having finite distance for that pair. Note that ChromLand requires exactly one distance per landmark-vertex pair by design, regardless of the dataset; we thus avoid to report its results.

The results in Table 2 confirm that the space required by the naïve approch is exponential in the numer of labels (never less than $2^{|L|-1}$). By contrast, for a smaller numbers of labels (i.e., up to 7-8 labels), the distances stored by PowCov roughly follow a linear trend. This is especially observable in the real datasets. For larger numbers of labels, the trend becomes more or less quadratic. In any case, the saving with respect to the exponential space required by the naïve approach is evident regardless of the labels, up to 95% and 87% on real and synthetic datasets, respectively. This confirms one of the major claims about our PowCov index: the number of SP-minimal label sets remains small enough in practice so to guarantee a significant space saving.

Indexing time. In Table 3 we report the indexing time required by our indexes averaged over the number of landmarks. We recall that PowCov is mainly designed to handle up to a few tens of labels (due to its intrinsic exponential-time index-building complexity), while no limitation in the number of labels affects ChromLand. For this purpose, results on synthetic datasets are reported up to 10 and 100 labels, for PowCov and ChromLand, respectively.

As far as PowCov, a major goal here is to compare the indexing time achieved by employing the proposed TraversePowerset algorithm to the time needed by the TraversePowerset-Brute-Force, so to validate the effectiveness of the pruning rules defined in Section 3.2.2. The results reported in the table show that our pruning rules indeed allow for significantly reducing the time of the brute-force method, up to around 70%. The speed-up increases as the number of labels increases: this is particularly appealing because it means that a larger speed-up is achieved in those cases that require more time due to a larger number of labels.

As expected, ChromLand indexing times are not really affected by the labels, thus testifying that ChromLand is a valid option when the number of labels is too large for PowCov. Particularly, the times are roughly constant for small/moderate numbers of labels, while becoming decreasing when the number of labels gets larger (i.e., ≥ 20). The motivation is the single color assigned to each landmark: the larger the number of labels in the graph, the smaller the set of vertices reachable with mono-chromatic paths, and, then, the smaller the time required by any label-constrained SSSP that is performed to build the index.

5.2 Query evaluation

Table 4 summarizes the query-processing results obtained by our indexes. Due to limited space, we focus here only on real datasets. For each proposed indexing method we report: (i) the absolute and relative errors of the estimated distances with respect to the exact distances, averaged over the number of all queries evaluated; (ii) the fraction of queries for which the index returns the *exact answer*; (iii) the fraction of queries for which the index mistakenly returns an infinite distance (false negatives); (iv) the *speed-up factor* with respect to exact shortest-path distance computation, averaged over

Table 2: Index sizes: average number of distances stored per landmark-vertex pair. The last line reports on how much (in percentage) the sizes of the naïve approach are reduced by PowCov.

	real datasets					*synthetic datasets (varying $\|L\|$)*						
index	BioGrid ($\|L\|$=7)	BioMine ($\|L\|$=7)	String ($\|L\|$=6)	DBLP ($\|L\|$=8)	YouTube ($\|L\|$=5)	4	5	6	7	8	9	10
PowCov	5.79	3.88	2.01	8.63	4.72	9.12	14.73	24.35	39.09	60.36	92.19	123.7
Naïve	84.24	74.43	34.66	116.3	29.21	13.39	27.69	56.59	115.1	233.3	470.68	950.7
	93.1%	94.8%	94.2%	92.6%	83.8%	31.9%	46.8%	57%	66%	74.1%	80.4%	87%

Table 3: Indexing times: average time (secs) per single landmark. PowCov times refer to the TraversePowerset algorithm (Algorithm 2), while BruteForce times refer to the TraversePowerset-BruteForce algorithm (Algorithm 1). The last line reports on how much (in percentage) the times of BruteForce are reduced by PowCov.

	real datasets					*synthetic datasets (varying $\|L\|$)*											
index	BioGrid	BioMine	String	DBLP	YouTube	4	5	6	7	8	9	10	20	30	40	50	100
ChromLand	0.2	4.43	0.04	0.18	2.5	4.1	4.8	5.7	5.6	6	6.6	6.4	3.4	2.7	2.02	1.7	1.2
PowCov	5.8	156.2	0.59	14.6	20.2	20.1	41.6	90.8	192.4	398	833.1	1783	—	—	—	—	—
BruteForce	11.3	269.8	1.09	38	29.4	33.2	76.2	179.5	409.4	963	2124	5631	—	—	—	—	—
	48.9%	42.1%	45.9%	61.7%	31.1%	39.5%	45.4%	49.4%	53%	58.7%	60.8%	68.3%					

Table 4: Query-processing results of the proposed indexes on real datasets with varying the number of landmarks: average absolute error, average relative error, percentage of exact answers, percentage of false-negative answers, and speed-up factor with respect to exact distance computation.

	PowCov, BioGrid					PowCov, BioMine					PowCov, String					PowCov, DBLP					PowCov, YouTube				
#landmarks	40	80	120	160	200	100	200	300	400	500	100	200	300	400	500	40	80	120	160	200	40	80	120	160	200
absolute error (avg)	2.02	0.87	0.69	0.65	0.35	1.07	0.91	0.81	0.78	0.58	1.19	1.06	0.88	0.87	0.72	1.44	0.97	0.68	0.59	0.45	0.46	0.38	0.34	0.32	0.3
relative error (avg)	0.44	0.2	0.16	0.15	0.09	0.31	0.27	0.25	0.24	0.18	0.38	0.35	0.29	0.3	0.24	0.24	0.16	0.11	0.1	0.08	0.28	0.24	0.22	0.21	0.2
exact answers (%)	20.8	45.1	57.9	61	82.6	33.2	41.0	46.8	48.3	62.3	32.3	38.2	48.2	51.7	58	20.5	35.9	50.2	55.9	64.7	56.6	63.2	67.1	69.4	70.1
false negatives (%)	0.33	0.33	0.33	0.33	0.33	0.004	0.003	0.003	0.003	0.003	34.6	24.1	19.8	14.6	12	0	0	0	0	0	0	0	0	0	0
speed-up factor	351	225	199	203	233	3 696	1 952	1 382	999	982	6 758	6 585	5 667	3 921	3 090	29	16	15	17	19	1 649	1 173	966	899	882

	ChromLand, BioGrid					ChromLand, BioMine					ChromLand, String					ChromLand, DBLP					ChromLand, YouTube				
#landmarks	40	80	120	160	200	100	200	300	400	500	100	200	300	400	500	40	80	120	160	200	40	80	120	160	200
absolute error (avg)	2.27	1.55	1.27	1.07	0.97	2.34	1.94	1.84	1.8	1.76	3.24	2.58	2.72	2.65	2.69	5.69	4.72	3.84	3.37	3.06	0.86	0.63	0.55	0.49	0.46
relative error (avg)	0.46	0.31	0.25	0.21	0.19	0.63	0.52	0.5	0.49	0.48	0.94	0.79	0.81	0.79	0.79	0.93	0.77	0.63	0.55	0.5	0.49	0.37	0.33	0.3	0.28
exact answers (%)	5.9	18.4	25	31.9	35.7	9	12	13.9	15	16	10.3	9.6	11.6	13.2	15.5	1.5	2.9	5.9	7.7	9.5	26.2	41.9	47.6	53.1	56
false negatives (%)	3.88	3.77	2.33	2.33	2.29	0.002	0.001	0.001	0.001	0.001	45	39.3	20.2	7.84	0.23	19.1	18.6	18.4	17.8	17.6	0.04	0.04	0.04	0.04	0.04
speed-up factor	344	123	58	32	23	4 073	1 429	616	435	270	2 274	1 844	1 621	1 573	1 352	54	16	7	4	3	2 257	881	466	295	220

all queries. Particularly, we measure the speed-up factor with respect to two baselines: the well-established bidirectional Dijkstra's algorithm [12] (adapted to the label-constrained context by ignoring the edges whose label is not in the query label set C)[3] and the technique by Rice and Tsotras [23]. We always report speed-up with respect to the fastest of such baselines. Somehow surprisingly, in our experiments, bidirectional Dijkstra is often more efficient than the method by Rice and Tsotras. A possible explanation is that the technique by Rice and Tsotras is mainly designed for road networks, whose special characteristics can be very different than those encountered in other real edge-labeled graphs.

We study the performance varying the number k of landmarks from $\log(n)$ to $\log^2(n)$ for each dataset. For the sake of consistency, we however show results over a common range of $[40, 200]$ (with step 40) for the smaller datasets (i.e., BioGrid, DBLP, YouTube), and $[100, 500]$ (with step 100) for the larger datasets (i.e., BioMine, STRING).

As expected, PowCov is the most accurate method on all datasets. Using $k = \max$ (i.e., either $k = 200$ or $k = 500$), the average relative error of PowCov is never larger than 0.25, while being even < 0.09 on DBLP. The high accuracy of PowCov is confirmed by the percentage of exact answers, which is (again with $k = \max$) around 58% on String, around 60-65$\%$ on BioMine and DBLP, 70% on YouTube, and 83% on BioGrid. The false-negative rate is very small as well: even zero on DBLP and YouTube, while less than 1% on BioGrid and BioMine.

Even being less accurate than PowCov, the ChromLand index reveals to be effective as well. For instance, on BioGrid and YouTube, the ChromLand average error is around 0.2 and around 0.3 (only 0.11 and 0.08 far from the error achieved by PowCov respectively), and the percentage of exact answers averaged over all datasets is 30%. Also, the false-negative rate remains small, e.g., almost zero on YouTube and less than 4% on BioGrid.

As a general observation, we remark that the accuracy of the proposed indexing methods does not depend on the diameter of the input graph: for instance, the error of both the proposed indexes remains very small on both BioGrid that has diameter 18 and YouTube that has diameter of only 6. This attests the generality of our methods to be applied to both graphs having large diameter and graphs where the *small-world phenomenon* prevails.

While the speed-up clearly depends on the number of landmarks, it is however significant in all the cases: 1-2 orders of magnitude on the smaller datasets (BioGrid, DBLP, YouTube), and even *three orders of magnitude* on the larger datasets (BioMine, STRING). The latter result is particularly relevant, as it testifies that the usefulness of the proposed indexing methods is even increasing as the size of the input graph increases. ChromLand reflects the quadratic complexity of the query processing with respect to the number of landmarks. Instead, the speed-up factor of PowCov is in general not monotone with the number of landmarks: on some datasets, such as BioGrid and DBLP, after a drop around $k = 120$, the speed-up grows again for larger k. The motivation of this behavior is the following. We recall that the query processing time in PowCov mainly depends on the (average) size of SP-minimal sets stored in the index. Thus, the results observed suggest that the number of such SP-minimal sets tends to be small and even to decrease as the landmarks increase. A major claim about PowCov is therefore supported: *the number of* SP*-minimal sets is in practice small enough to ensure fast query processing*.

[3] As all the graphs used in our experiments are unweighted, in our implementation we actually use bidirectional BFS.

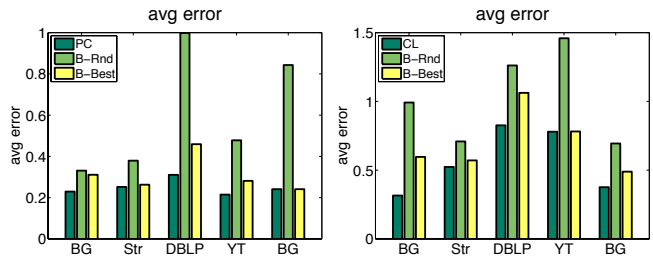

Figure 6: Landmark selection on real datasets: PowCov (left) and ChromLand (right) vs. the B-Rnd and B-Best baselines.

5.3 Landmark selection

We next assess the importance of smart landmark selection against random selection and other well-known baselines. For both PowCov and ChromLand we compare the proposed landmark-selection strategies introduced in Sections 3.3 and 4.3 against the results obtained by the same PowCov and ChromLand indexes when equipped with landmarks selected according to some baselines. Besides the most obvious baseline of selecting landmarks at random, we consider as smarter baselines landmarks having the best (approximated) *betweenness-centrality* scores, landmarks selected by the TopDegreeMVC algorithm for k-MAX-VERTEX-COVER [7] (which corresponds to selecting as landmarks the vertices with *highest degree*), as well as landmarks selected from a *vertex cover* of the input graph (according to either betweenness centrality or degree). As far as ChromLand, we consider two variants of each baseline: one where the landmark colors are assigned at random, and another one where the color of each landmark is set equal to the majority color among its incident edges. For the sake of brevity, we report results about the most obvious baseline (totally random selection, denoted B-Rnd) and the best one over all the other baselines (denoted B-Best).

The results of this evaluation are shown in Figure 6, where we report the relative errors achieved by both PowCov and ChromLand on all real datasets, averaged over all queries, with number of landmarks $k \in [40..200]$ (BioGrid, DBLP, YouTube) or $k \in [100..500]$ (BioMine, String). It is easy to see how employing our landmark selection leads to evident accuracy improvements with respect to the baselines, for both PowCov and ChromLand. Using our PowCov landmarks reduces the average errors of the B-Rnd and B-Best baselines up to 71% (on YouTube) and 32% (on String), respectively. A similar behavior is observed with ChromLand. Here, the errors of the B-Rnd and B-Best baselines are reduced up to 68% and 47% (on BioGrid), respectively.

6. CONCLUSIONS

We addressed the problem of fast online approximation of label-constrained shortest-path distance queries in edge-labeled graphs. We presented two landmark-based indexes that tradeoff between storage vs. accuracy/efficiency: the first index is faster and more accurate, but it takes more space and is more expensive to build. Experiments on synthetic and real-world datasets revealed the high performance of our indexes. Both indexes achieve a speed-up of up to 3 orders of magnitude compared to exact shortest-path distance computation, while keeping the approximation error small.

7. REFERENCES

[1] P. Agrawal, V. K. Garg, and R. Narayanam. Link label prediction in signed social networks. In *IJCAI*, 2013.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*, 1994.

[3] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. *SIAM J. Comput.*, 2004.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 2003.

[5] B. Boden, S. Günnemann, H. Hoffmann, and T. Seidl. Mining coherent subgraphs in multi-layer graphs with edge labels. In *KDD*, 2012.

[6] F. Bonchi, A. Gionis, F. Gullo, and A. Ukkonen. Chromatic Correlation Clustering. In *KDD*, 2012.

[7] G. Cornujols, G. L. Nemhauser, and L. A. Wolsey. Worst-case and probabilistic analysis of algorithms for a location problem. *Operations Research*, 1980.

[8] W. Fan, J. Li, S. Ma, N. Tang, and Y. Wu. Adding regular expressions to graph reachability and pattern queries. In *ICDE*, 2011.

[9] S. P. Fekete, T. Kamphans, and M. Stelzer. Shortest paths with pairwise-distinct edge labels: Finding biochemical pathways in metabolic networks. *CoRR*, abs/1012.5024, 2010.

[10] M. Fire, L. Tenenboim, O. Lesser, R. Puzis, L. Rokach, and Y. Elovici. Link prediction in social networks using computationally efficient topological features. In *SocialCom/PASSAT*, 2011.

[11] R. Geisberger, P. Sanders, D. Schultes, and D. Delling. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *WEA*, 2008.

[12] A. V. Goldberg and C. Harrelson. Computing the shortest path: A* search meets graph theory. In *SODA*, 2005.

[13] A. Gubichev, S. J. Bedathur, S. Seufert, and G. Weikum. Fast and accurate estimation of shortest paths in large graphs. In *CIKM*, 2010.

[14] M. A. Hasan, V. Chaoji, S. Salem, and M. Zaki. Link prediction using supervised learning. In *SIAM SDM Workshops*, 2006.

[15] D. Hochbaum. *Approximation algorithms for NP-hard problems*. 1997.

[16] R. Jin, H. Hong, H. Wang, N. Ruan, and Y. Xiang. Computing label-constraint reachability in graph databases. In *SIGMOD*, 2010.

[17] B. P. Kelley, B. Yuan, F. Lewitter, R. Sharan, B. R. Stockwell, and T. Ideker. Pathblast: a tool for alignment of protein interaction networks. *Nucleic Acids Research*, 2004.

[18] J. Kleinberg, A. Slivkins, and T. Wexler. Triangulation and embedding using small sets of beacons. In *FOCS*, 2004.

[19] J. Leskovec, A. Singh, and J. M. Kleinberg. Patterns of influence in a recommendation network. In *PAKDD*, 2006.

[20] J. J. McAuley and J. Leskovec. Learning to discover social circles in ego networks. In *NIPS*, 2012.

[21] M. Potamias, F. Bonchi, C. Castillo, and A. Gionis. Fast shortest path distance estimation in large networks. In *CIKM*, 2009.

[22] M. Qiao, H. Cheng, L. Chang, and J. X. Yu. Approximate shortest distance computing: A query-dependent local landmark scheme. In *ICDE*, 2012.

[23] M. Rice and V. J. Tsotras. Graph indexing of road networks for shortest path queries with label restrictions. *PVLDB*, 2010.

[24] A. D. Sarma, S. Gollapudi, M. Najork, and R. Panigrahy. A sketch-based distance oracle for web-scale graphs. In *WSDM*, 2010.

[25] C. Sommer. Shortest-path queries in static networks, 2012. submitted, available at: `www.sommer.jp/spq-survey.pdf`.

[26] L. Tang, X. Wang, and H. Liu. Community detection via heterogeneous interaction analysis. *DAMI*, 2011.

[27] M. Thorup and U. Zwick. Approximate distance oracles. In *STOC*, 2001.

[28] K. Tretyakov, A. Armas-Cervantes, L. García-Bañuelos, J. Vilo, and M. Dumas. Fast fully dynamic landmark-based estimation of shortest path distances in very large graphs. In *CIKM*, 2011.

[29] K. Xu, L. Zou, J. X. Yu, L. Chen, Y. Xiao, and D. Zhao. Answering label-constraint reachability in large graphs. In *CIKM*, 2011.

[30] M. Zaslavskiy, F. Bach, and J.-P. Vert. Global alignment of protein-protein interaction networks by graph matching methods. *Bioinformatics*, 2009.

[31] A. Zhang. *Protein Interaction Networks: Computational Analysis*. Cambridge University Press, 2009.