Declarative Cartography under Fine-Grained Access Control

Thomas Jensen*
June, Danske Bank
Copenhagen, Denmark
vxf838@alumni.ku.dk

Marcos Antonio Vaz Salles†
University of Copenhagen
Copenhagen, Denmark
vmarcos@di.ku.dk

Michael Vindahl Bang*
June, Danske Bank
Copenhagen, Denmark
tqg432@alumni.ku.dk

ABSTRACT

Visualization of spatial data is of increasing importance in science and society, but opens up justified concerns about data privacy and security. A classic methodology for cartography through generalization is data selection; however, data selection can be challenging under security constraints for two main reasons. First, individual records are kept in the visualization, so a data security approach such as access control needs to be put in place to avoid leakage of information about protected records to unauthorized parties. Second, it can be computationally hard to pick out records from a large spatial dataset so as to create an aesthetically pleasing visualization respecting user constraints and optimization goals. The latter expense can get compounded by the need to additionally respect access control restrictions.

This paper presents a way to integrate label-based access control into an existing technique for declarative cartography termed global selection. Through a set of theorems and new algorithms, we demonstrate that we can reuse derivation and resolution of record conflicts when computing global selections across access roles in a security hierarchy. In experiments with realistic datasets, the runtime of the best among these new methods achieves an improvement of up to 2x-5x compared with repeatedly computing the global selection in medium-to-large security hierarchies.

1 INTRODUCTION

Motivation. Geospatial visualizations such as maps are of crucial value in geographic information science [14], and becoming increasingly important in organizations with the trends towards spatial computing [38] and Internet of Things (IoT) [40]. Data security and privacy in these domains is of great concern [10, 20]. For example, a recent survey shows that 75% of expert respondents believe IoT cybersecurity is of top priority or important, while only 16% believe their organizations are ready for the security challenges of IoT [20].

Cartographic generalization techniques are commonly employed to create geospatial visualizations [19]. Among these techniques, an approach that particularly challenges data security is that of data selection. On the one hand, data selection techniques can be computationally expensive, since methods need to respect complex user-defined visualization constraints and/or optimize for user-defined goals [24, 25, 32, 35]. On the other hand, data selection by definition picks out individual records from a large dataset, which can compromise privacy and security constraints. Other than data transformations applied to (groups of) records [31], a valid data security approach in creating such visualizations is fine-grained access control [23, 33, 34, 42], where selected records are only shown to authorized parties. It remains, however, unclear how to efficiently combine cartographic selection methods with access control, a problem that we study in this paper.

Examples. Figures 1 to 3 illustrate the problem of combining a cartographic data selection method with fine-grained access control. The figures show three different visualizations of a dataset containing information about airports. The colors red and black represent the access level of an airport. For example, the airports could be military airports where red airports are top-secret and black airports are secret. A user with top-secret access clearance has access to both top-secret and secret airports. It is apparent that by showing the full dataset (Figure 1), the data points become too cluttered and we can hardly distinguish one airport from another. By using the *global selection* method of [24, 25] for declarative cartography, we can define a constraint that no two airports may appear within a certain pixel distance of each other, thus selecting a subset of the whole dataset. Now, we can compute the global selection for a top-secret user (Figure 2) and for a secret user (Figure 3).

It is intuitive that there must be a lot of commonality between the two visualizations since the secret airports form a subset of the data accessible to a top-secret user. However, to avoid leakage, we need to compute the global selection for the two visualizations independently. It seems rather inefficient to identify airports within a certain distance of each other twice from scratch. Ideally, we would like to reuse the computation of the global selection with top-secret access in deriving the global selection with secret access.

There are countless other examples where users have access to different subsets of the whole data. Consider social networks where a user can restrict access to personal information for some of her friends and family, or a consultant in an agricultural extension service who can access data from several farmers.[1] In these scenarios, we have an inherent notion of a security hierarchy. Additionally, we expect a certain commonality between the data accessible to different users. For example, the consultant and one of her clients will both have access to data concerning that farm, i.e., the data accessible to the client is a subset of that accessible to the consultant.

*Work performed while studying at University of Copenhagen.

†Work partially performed in the context of the Future Cropping partnership [13], supported by Innovation Fund Denmark.

SSDBM '18, July 9–11, 2018, Bozen-Bolzano, Italy

ACM ISBN 978-1-4503-6505-5/18/07…$15.00
https://doi.org/10.1145/3221269.3232012

[1] The problem addressed in this paper was inspired by discussions within the FutureCropping project [13], which deals exactly with an agricultural extension service scenario.

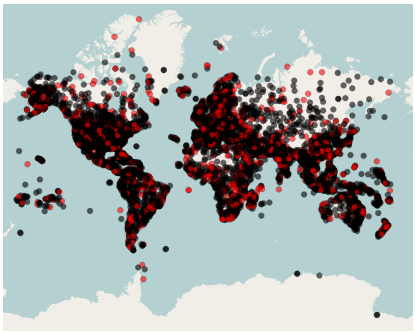

Figure 1: Openflights Airport dataset (7K points). Red access label dominates black.

Figure 2: Airports with proximity constraint with access to both red and black airports.

Figure 3: Airports with proximity constraint with access only to black airports.

Despite this notion of inclusion of data in security hierarchies and the need for cartographic data selections in geospatial visualizations, to the best of our knowledge no previous work has designed efficient mechanisms to exploit data inclusion in computing global selections under different access roles.

Problem. Since we cannot show any top-secret airports to a person with only secret access, we could simply remove any top-secret airports from Figure 2 to produce a visualization for such a person. However, this would introduce two problems:

(a) The visualization no longer maximizes the information given to the user. Any top-secret airport could have led to the elimination of a nearby secret airport, and the secret airport no longer appears in the output.
(b) By having such "gaps" in the map, a user with secret access might deduce where top-secret airports are located.

Had we simply removed the red airports from Figure 2 to generate Figure 3, the visualization would show far fewer points. For example, we would not see any airport in New Zealand or Central/Eastern Europe. As such, we may be able to deduce that there might be a top-secret airport in these areas. The goal therefore is to always generate an output for the secret user equal to the global selection on just the secret airports, i.e., as if the top-secret airports did not exist. In summary, we need to preserve quality in the output as well as not compromise the integrity of a given security hierarchy. However, we wish to do so without having to recompute the global selection from scratch for each access role.

Offline versus online computation. Much previous research in data visualization focuses on interactivity [17, 21, 22]. Recent methods, such as dynamic reduction [4], IDEAs [9], NanoCubes [27] or imMens [28], employ sampling or aggregation operations, which are either computationally inexpensive or parallelizable. However, data selection is often expensive enough that it will be infeasible to compute "live" [24, 25, 32, 35]. Luckily, offline computation of selections can still be used to prepare data for visualization. For example, city labels and road names, among others, can be selected in a pre-processing step for later display in an interactive map [24]. In this paper, we focus on this pre-computation step, and how it gets affected by the use of fine-grained access control in the database.

Contributions. This paper makes the following contributions:

(1) Motivated by the notion of commonality of data between access roles, we formalize and prove a number of theorems that can be used to piece together a global selection for a lower access level from the one of a higher access level. These results delimit the conditions in which the latter can be done safely, covering a wide array of user-defined visual constraints employed with global selections.
(2) Based on our theoretical results, we present new algorithms to efficiently compute global selections under access control.
(3) In experiments with realistic datasets, we compare the performance of our algorithms to the naïve approach of computing the global selection for every role in a security hierarchy. Our best method achieves runtime improvements of 2x-5x in medium-to-large security hierarchies.

For concreteness, this paper focuses on label-based access control [34] and the global selection method of [24, 25] for declarative cartography, which we briefly review in Section 2. Our theoretical results and algorithms are presented in Section 3 and experiments in Section 4. We discuss related work in Section 5.

2 BACKGROUND

2.1 Access Control Models

Fine-Grained Access Control Methods. We focus on the approach of fine-grained access control, which has been studied extensively [23, 33, 34, 42]. Since we need an access control method where access roles can *include* the data of other roles, we have chosen to use a label-based access control model similar to what is presented by Rjaibi and Bird [34]. This method seamlessly integrates a hierarchical structure between access roles by introducing the concept of *dominance*, which we formalize below.

Label-Based Access Control. We define a label $l \in \mathbb{L}$ to be a string, where $\mathbb{L}$ is the set of all possible labels. A security hierarchy is a partial order $\preceq$ on a finite set of labels. We denote the domain of all possible partially ordered sets of labels as $\mathbb{A}$.

In the remainder, we assume an input relation I is always subject to a security hierarchy $A = (\{l_1, l_2, ..., l_n\}, \preceq) \in \mathbb{A}$. We denote the tuples visible to label l_i in an input relation I as $I^{l_i} = \sigma_{label \preceq l_i}(I)$. I^{l_i} is read as I *subject to label* l_i. We say l_j *dominates* l_i if $l_j \succeq l_i$ and thus $I^{l_j} \supseteq I^{l_i}$. Using this terminology, we can think of I subject to l_i to be the subset of tuples from I that are visible to a user who has access to label l_i.

Representation of a Security Hierarchy. A security hierarchy as defined above can be represented by different data structures. The example of top-secret and secret labels consists of a linear hierarchy and can be represented by a simple list. However, many security hierarchies are more complex than that. For example, company structures or consultant relations lead to hierarchies that are more complex trees or even DAGs.

To capture this variety in security hierarchies while maintaining efficiency, our implementation employs two different encodings for security hierarchies: one for trees [7] and one for DAGs [43]. Both encodings can be implemented in a relational DBMS. The first encoding method is an approach where the pre/post plane of the security hierarchy defines the labels [7]. While the first encoding method is restricted to trees, the second encoding method applies to DAGs and is based on prime number factorization [43]. The second method is less effective in the sense that labels become huge very fast as the security hierarchy grows; however, the encoding is sufficient to cover our experiments with DAGs in Section 4.

The work in this paper does not mandate either of these encodings. We could encode the security hierarchy using other methods as long as the dominance between any two labels is well-defined. For our implementation, we assume updates to the security hierarchy itself are rare, and can easily be handled by periodically rebuilding the data structure chosen for representation.

2.2 Global Selections

Previous work introduced the concept of database-integrated global selections for declarative cartography [24, 25]. We summarize and revisit this work in this section, providing formal definitions that support our work of bringing access control to global selections.

Global Selection Concepts. Global selection is a technique to choose relevant records from a dataset by evaluating the data as a whole. The two building blocks of a global selection are a set of *constraints* and a *weighting function*. Constraints specify restrictions that must be respected by sets of records in the input. If a set of records matches the restrictions in a constraint, we say that the records in the set form a *conflict*. For the dataset shown in Figure 1, we show the output of a global selection with a proximity constraint for a user with access to all data in Figure 2 and for a user with access to only the black airports in Figure 3. A proximity constraint states that no two records should be closer than a pixel distance d to one another. Given such a constraint, a set of two sufficiently close records forms a conflict. The weighting function determines the importance, or weight, of different records. In the example of Figure 1, the weight is given by the number of departures of an airport, which acts as a proxy for its importance.

The processing of a global selection consists of two basic steps: *conflict derivation* and *conflict resolution*. Conflict derivation calculates the sets of records forming conflicts by applying the constraint definitions over an input relation. Conflict resolution eliminates records from the input relation so as to resolve all conflicts and thus satisfy all constraints, while attempting to maximize the weight of the resulting output set of records.

Global Selection Formalization. To formalize the notion of global selection, we need to first define how weighting functions and constraints apply to the tuples of a relation. A weighting function w is simply a mapping from a given tuple to a non-negative real weight. A constraint κ is a mapping from an input relation I to a finite set of pairs $\{(\tau, R_1), (\tau, R_2), \ldots, (\tau, R_k)\}$, where τ is a non-negative integer threshold and each $R_i \subseteq I$ is a conflict. Each pair encodes the limit on the maximum number of records τ that are allowed by the constraint to survive out of the records in the conflict R_i. For example, in a proximity constraint we would have $\tau = 1$ across all pairs, and each R_i would be a set consisting of two records from the input relation I that are sufficiently close to each other. This constraint is schematically illustrated in Figure 4. Following the model of declarative cartography [25], we assume for this paper that τ is constant for all R_i within a single constraint.

Based on constraints, we can formalize conflict derivation.

Definition 2.1 (Conflict Derivation Function). A conflict derivation function CD is a mapping from an input relation I and a set of constraints K such that:

(1) $CD(I, K) = \kappa(I)$, if $K = \{\kappa\}$;
(2) $CD(I, K) = CD(I, \{\kappa_1\}) \cup CD(I, \{\kappa_2\}) \cup \ldots \cup CD(I, \{\kappa_n\})$, if $K = \{\kappa_1, \kappa_2, \ldots, \kappa_n\}$.

CD derives all conflicts along with their respective thresholds for all given constraints. Thus, the output of CD is a set of pairs $\{(\tau_1, R_1), (\tau_2, R_2), \ldots, (\tau_m, R_m)\}$. For convenience, we define the *conflict set* C produced by CD as $C = \{R_1, R_2, \ldots, R_m\}$. Conflict sets allow us to reason on the records included in conflicts without being bothered by the thresholds imposed.

Note that the *set of constraints* K as well as the *weighting function* w are defined by the user. From these input definitions, we can construct the conflict derivation function as above. In addition, we can define a conflict resolution function.

Definition 2.2 (Conflict Resolution Function). A conflict resolution function CR^w is a mapping from an input relation I and the set of pairs produced by a conflict derivation function application $CD(I, K)$ to an output relation $O \subseteq I$ such that:

(a) O contains at most τ_i surviving records for each of the conflicts in the pairs $(\tau_i, R_i) \in CD(I, K)$;
(b) the sum of the record weights given by weighting function w in O is maximal w.r.t. all such qualifying relation instances.

For the proximity constraint example in Figure 4, conflict resolution yields the result shown in Figure 5, assuming t_2 has higher weight than t_3, and t_4's weight is lower than the combined weight of t_5 and t_6. Conflict resolution is equivalent to the weighted set multi-cover problem and thus NP-hard [24, 25]. In spite of this, previous work has shown that effective heuristics can be applied to conflict resolution in realistic map making scenarios [25].

Now, we can formalize a global selection as follows.

Definition 2.3 (Global Selection Operator). Given an input relation I, a weighting function w, and a set of constraints K, we define a global selection as $\Sigma_K^w(I) = CR^w(I, CD(I, K))$.

In contrast to [24, 25], we limit our attention to single-scale global selections, i.e., we do not consider visualizations where multiple zoom levels are calculated in a manner sensitive to access control. A simple strategy to deal with multi-scale global selections is to abstract these operations as a sequence of single-scale global

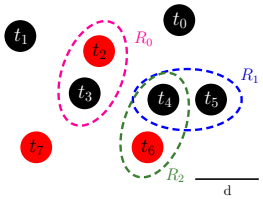

Figure 4: Conflicts under proximity constraint with distance d.

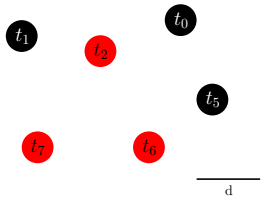

Figure 5: Global selection for the red label, eliminating t_3 and t_4.

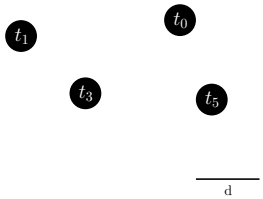

Figure 6: Global selection for the black label, eliminating t_5.

selections, and employ the methods developed in this paper at each scale. Notwithstanding, single-scale global selections are of independent interest. Broadly speaking, many infographics that show a selection of data can be made with a single-scale global selection. Some examples hereof are maps as used in articles and books, or visualizations in newspapers and scientific research papers.

2.3 Monotonicity

We aim to take advantage of the commonality between datasets of two different access labels to derive the global selection for one access label using the conflicts and outputs of another. However, achieving this goal under arbitrary conflict derivation functions may be infeasible. For example, if conflicts were decided at random in each global selection, we could not use the conflicts from a global selection for one access label in the global selection of another.

Gladly, common conflict derivation functions exhibit properties that we can exploit. In particular, it is unusual that the set of tuples in conflict will shrink if *more* tuples are added to the input. Consider the case of the proximity constraint. If more tuples are added to the input set, there is a chance that now new tuples will be too close to existing tuples or that added tuples will be too close to one another. However, the distances between existing tuples do not change. As another example, consider a constraint limiting the density of records in space. As we add more records to the input, the density can only increase.

The property in these examples is captured by requiring the conflict derivation function CD from Definition 2.1 to be *monotonic* in the conflict sets produced. To formalize this notion, we need to introduce conflict set containment.

Definition 2.4 (Conflict Set Containment). Given two conflict sets C_n and C_m, C_m contains or is equal to C_n, denoted $C_m \sqsupseteq C_n$, iff. $\forall R_i \in C_n, \exists R_j \in C_m : R_i \subseteq R_j$.

In a manner similar to query monotonicity [1], given a set of constraints K and for any input relations $I_n, I_m \subseteq I$ with $I_m \supseteq I_n$, we say CD is *monotonic* whenever if we take the conflict sets C_n and C_m induced by $CD(I_n, K)$ and $CD(I_m, K)$, then $C_m \sqsupseteq C_n$. In other words, adding a tuple to an input relation may only grow the set of tuples in a conflict. Intuitively, this implies that the constraints used are such that the conflict between two tuples cannot rely on the absence of another tuple, which is a natural restriction in practice.

Consider again the proximity constraint as illustrated in Figure 4. Suppose t_4 were inexistent and then added to the input. If t_4 would have no other existing points within d distance of it, no new conflicts would be generated. However, t_4 is within d distance of existing points. So we generate one new conflict for each such qualifying points, namely t_5 and t_6. Conflict derivation using the proximity constraint is monotonic.

We also consider a stronger notion of monotonicity where adding a tuple may only grow a conflict by that particular tuple, namely *strong monotonicity*. Suppose we have an input relation $I = \{a, b, c\}$ and conflict derivation such that a and b are in a conflict R. We then add tuples d and e to I. According to strong monotonicity, R may at most grow by either d, e, or both. Additionally, a new conflict between d and e could be introduced. By contrast, were we to use a conflict function with the weaker notion of monotonicity, c could end up in R when including d in I. Also, d, e, or both could end up in a new conflict along with any of a, b, or c.

In the proximity constraint, observe that *if* we generate a new conflict, it may contain not only the added but also an existing tuple. Therefore, the proximity constraint is not strongly monotonic, even though it is monotonic. As we discuss in Section 3.4, the density constraint alluded to above is strongly monotonic.

3 ACCESS CONTROL FOR GLOBAL SELECTIONS

In this section, we study global selections subject to access control. We start with a brief overview of the problem (Section 3.1), followed by the *naïve approach* to its solution (Section 3.2). We then present a set of results that allows us to compute global selections with access control for monotonic conflict derivation functions (Section 3.3), followed by results for a special case common in practice, namely functions yielding disjoint conflicts (Section 3.4).

3.1 Problem Overview

In essence, given a security hierarchy for an input relation, we would like to compute the global selection for every label in the security hierarchy. To see why this is so, suppose each user is assigned an access label. Intuitively, we want the result to be equal to the scenario where every user runs the global selection on the subset of tuples from the input relation she has access to.

To consider global selections in the context of access control, we need to introduce a few notations. R^l is the conflict where each tuple is subject to security label l. C^l is thus the set of conflicts for a given set of constraints, where each conflict is subject to l. Finally, we denote by O^l the output from applying the global selection operator Σ^w_K on an input relation I^l. In general, a set subject to a label is, unless otherwise specified, the selection of all records dominated by that label (recursively applied if the set is a set of sets). We denote a tuple t^l if the label of the tuple is l (i.e., $t.label = l$).

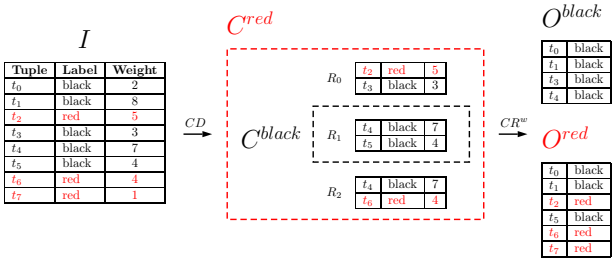

Figure 7: Conceptual evaluation of the global selection from Figure 4 for two access labels, *red* and *black*, where *red* dominates *black*.

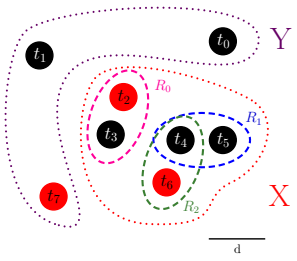

Figure 8: Constrained (X) and unconstrained (Y) tuples in Figure 4.

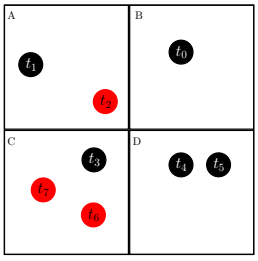

Figure 9: Tiles represent conflicts generated by a density constraint.

In addition, we adopt a flat relational representation for a set of conflicts C, in line with [24, 25]. Suppose the input relation I has attributes $\langle i_1, i_2, ..., i_p \rangle$ where the access label is assumed to be among the attributes. The set of conflicts C then has attributes $\langle i_1, i_2, ..., i_p, cid, w, \tau \rangle$, where cid is the ID from the conflict the tuple appears in, w is the weight of the tuple, and τ is the maximum number of tuples that may survive the conflict with conflict ID given by cid. Note that a tuple can be present in multiple conflicts and thus appear several times in C, but with different cid. In other words, the addition of cid is how we flatten the structure of C (a set of sets) to a single relation without loss of information. The output relation O has an identical schema to I.

Consider again the example of Figures 4 to 6, which illustrates a global selection for a security hierarchy of two labels. Figure 7 shows how the global selection is computed using the notations introduced above. We observe that since t_0, t_1, and t_7 are in no conflicts, they are automatically included in O^{red}. The black label cannot see the red tuples. Therefore, t_3 is not in any conflicts seen through the eyes of the black label and is included with t_0 and t_1 in O^{black}. Note that no red tuples can end up in O^{black} because they simply do not exist for black. The only conflict subject to the black label is R_1. CR^w will try to maximize the aggregated weight of the output and thus picks t_4 over t_5. Subject to the red label, we have three conflicts. Notice that t_4 is in both R_1 and R_2. Were CR^w to pick t_4 (the tuple with the highest weight), it must eliminate both t_5 and t_6. However, the combined weight of t_5 and t_6 is greater than the weight of t_4. Therefore, CR^w eliminates t_4 and picks t_5 and t_6. This example shows how t_5, a tuple visible to black, can end up in the output of red (the dominating label) without ending up in the output of black.

3.2 Naïve Approach

Algorithm 1 : ***Naive***
Naïve Approach to Global Selection with Access Control

Require: Global selection Σ_K^w
Require: Input relation I
1: **for** Label l in L **do**
2: $O^l \leftarrow \Sigma_K^w(I^l)$
3: MATERIALIZE(O^l)
4: **end for**

The simplest way of computing the global selection for every label in a security hierarchy L is to iteratively take the input relation subject to each label $l \in L$ and compute the whole global selection from scratch. We call this method the *naïve approach*. This simple method, which is trivially correct and shown in Algorithm 1, is the baseline for all the algorithms to come; the approach we wish to beat in terms of running time.

As discussed in more detail in Section 4.2.1, our implementation reuses SQL for encoding the algorithms presented in the paper. The construct *MATERIALIZE* in Algorithm 1, in particular, corresponds to view materialization of its logical input relation.

3.3 Monotonic CD

In this section, we present results and algorithms that show that portions of the computation of a global selection for a given label can be reused for the computation of the same global selection for dominated access labels. In particular, reuse of derived conflicts of a global selection can be very effective in reducing computation time, since it is often the case that the computation of CD is expensive in practice. For example, with a proximity constraint, the computation of CD requires a spatial join, and this operation can dominate the running time of a global selection [25].

To reason about reuse, we first establish a lemma relating the conflict sets of a global selection with different labels in a security hierarchy, exploiting the notion of commonality and inclusion of accessible data between two labels.

Lemma 3.1 (Conflict Containment of Input). *Consider an input relation I and a security hierarchy $A = (\{l_1, l_2, ..., l_n\}, \preceq) \in \mathbb{A}$ where $l_j \succeq l_i$. $C^{l_j} \sqsupseteq C^{l_i}$ if CD is monotonic.*

Proof. Recall from Section 2.1 that $I^{l_i} \subseteq I^{l_j}$. Let $DIFF^{l_j,l_i} = I^{l_j} - I^{l_i}$. Assume CD is monotonic and we have computed C^{l_i}. Now, we add the tuples from $DIFF^{l_j,l_i}$ one at a time to the input. By our definition of monotonicity, adding any tuple to the input can only create new conflicts or increase the size of the existing conflicts. Therefore, after having added $DIFF^{l_j,l_i}$ to I^{l_i}, we end up with I^{l_j} and for all conflicts in C^{l_i} there now exists a conflict that is a superset of or equal to it (i.e., $\forall R^{l_i} \in C^{l_i}, \exists R^{l_j} \in C^{l_j} : R^{l_i} \subseteq R^{l_j}$). By Definition 2.4, C^{l_j} contains or is equal to C^{l_i}. □

For example, in Figure 7 where red dominates black, we see that C^{black} is contained within C^{red}. Furthermore, we now know that the set of conflicts for a label is contained within the set of conflicts for any dominating label. Motivated by this fact, we formalize when we can reuse the conflicts in such a case.

Theorem 3.2 (Cached Conflicts). *Let $A = (\{l_1, l_2, ..., l_n\}, \preceq) \in \mathbb{A}$ where $l_j \geq l_i$ and CD be strongly monotonic. $\forall R^{l_j} \in C^{l_j}$ let $D^{l_i,l_j} = \{x \in R^{l_j} : x.label \succ l_i\}$, then $R^{l_i} = R^{l_j} - D^{l_i,l_j}$.*

Proof. Let R^{l_i} be any conflict from C^{l_i}. Since CD is monotonic, we know by Lemma 3.1 that $C^{l_j} \sqsupseteq C^{l_i}$ and thus by Definition 2.4 that $R^{l_i} \subseteq R^{l_j}$. Because CD is also strongly monotonic, any tuple in R^{l_j} not in R^{l_i} must be exactly $D^{l_i,l_j} = \{x \in R^{l_j} : x.label \succ l_i\}$. Therefore, we can remove all tuples in D^{l_i,l_j} from R^{l_j} without removing any tuples also in R^{l_i} and thereby get R^{l_i}. □

Observation 1 (Dominated Conflict). *Let $A = (\{l_1, l_2, ..., l_n\}, \preceq) \in \mathbb{A}$ where $l_j \geq l_i$ and CD be strongly monotonic. $\forall R \in C^{l_j}$, if $\forall t \in R$, $t.label \preceq l_i$, then $R \in C^{l_i}$. This is the special case of Theorem 3.2 where $D^{l_i,l_j} = \emptyset$.*

Theorem 3.2 shows that when CD is strongly monotonic, we can reuse the conflict sets of dominating labels in the computation of the global selection for dominated labels. Observation 1 targets the situation where all tuples in a conflict are accessible to the dominated label. Let us again use the simple top-secret/secret security hierarchy as an example. Observation 1 says that if every tuple in a given conflict is secret in the top-secret global selection, then we can reuse the conflict in the secret global selection.

We are now able to extend the Naive algorithm using Theorem 3.2 such that we only compute the conflicts once and then reuse them for every label in the security hierarchy.

Algorithm 2 : ***CC***
Cached Conflicts using Theorem 3.2

Require: Set of constraints K
Require: Strongly-monotonic conflict derivation function CD
Require: Conflict resolution function CR^w
Require: Input relation I
1: $C \leftarrow CD(I, K)$
2: MATERIALIZE(C)
3: **for** Label l in L **do**
4: $\quad O^l \leftarrow CR^w(I^l, C^l)$
5: $\quad$ MATERIALIZE(O^l)
6: **end for**

Correctness Argument 1 (Algorithm 2). *Let CD be strongly monotonic. Then, $C^l = \sigma_{label \preceq l}(C)$ by Theorem 3.2 and we defined in Section 2.2 that $CD(I^l, K)$ is equal to C^l. Thus, we do not need to compute the conflicts using CD for any labels and still get the correct set of conflicts at each iteration by subjecting C to the current label. Recall from Section 2.2 that $\Sigma^w_K(I^l) = CR^w(I^l, CD(I^l, K))$. The result of $CR^w(I^l, C^l)$ at each iteration is therefore equivalent to that of line 2 in Algorithm 1.* □

Observation 2 (Conflicts in pairs). *Let $A = (\{l_1, l_2, ..., l_n\}, \preceq) \in \mathbb{A}$ where $l_j \geq l_i$ and CD be monotonic generating conflicts exclusively of size two where $\forall R \in C^{l_j}$, if $\forall t \in R$, $t.label \preceq l_i$, $R \in C^{l_i}$, then Theorem 3.2 holds.*

A consequence of Observation 2 is that we can also use the CC algorithm on some monotonic, but not strongly monotonic, CD functions. These functions should generate exclusively conflicts in pairs and when both tuples in a pair exhibit a dominated label, then the pair must appear as a conflict for the dominated label as well. We expect this case to be very useful in practice, since it is natural for constraints to calculate conflicts by join operations. For example, the proximity constraints falls into this category. Given that it is computed by a spatial join, conflicts at a given label do not disappear as more tuples become visible at a dominating label.

Theorem 3.2 allows us to reuse conflicts are contained within the conflict set of dominating labels; however, reuse is also possible for tuples that are not included in any conflict at all. If we look at Figure 4, we observe that t_0 and t_1 are examples of such tuples taking part in no conflicts under the red access label. Suppose now we only have access to the black tuples. By monotonicity, t_0 and t_1 must still be in no conflicts. For brevity, we call tuples in at least one conflict *constrained tuples*, and tuples in no conflicts at all *unconstrained tuples*. We now formalize reuse for the latter.

Theorem 3.3 (Unconstrained Tuples). *Let $A = (\{l_1, l_2, ..., l_n\}, \preceq) \in \mathbb{A}$ where $l_j \geq l_i$. Then $\forall t \in O^{l_j}$ where $t \notin \{x \mid \exists R \in C^{l_j} : x \in R\}$ and $t.label \preceq l_i$, $t \in O^{l_i}$.*

Proof. Let t be a tuple *not* in O^{l_i} that is in O^{l_j} where $t.label \preceq l_i$ and $t \notin \{x \mid \exists R \in C^{l_j} : x \in R\}$. t must have been in some conflict $R' \in C^{l_i}$ to get eliminated from O^{l_i}. We now have, $\nexists R \in C^{l_j}$ s.t. $R \supseteq R'$. That is a contradiction by Lemma 3.1. □

Note that Theorem 3.3 does not require a strongly monotonic conflict derivation function CD. We can thus use the theorem on a wider range of conflict derivation functions than Theorem 3.2. We can exploit Theorem 3.3 by identifying unconstrained tuples for a dominating label, and then directly adding these tuples to the output for all dominated labels. Thereby, we decrease the input size to both CD and CR^w. Algorithm 3 captures this idea.

Algorithm 3 : ***CU***
Unconstrained Tuples using Theorem 3.3

Require: User-defined set of constraints K
Require: Monotonic conflict derivation function CD
Require: Conflict resolution function CR^w
Require: Input relation I
1: $C \leftarrow CD(I, K)$
2: $X \leftarrow I \ltimes C$ /* Tuples in a conflict */
3: $Y \leftarrow I - X$ /* Tuples not in any conflict */
4: MATERIALIZE(X)
5: MATERIALIZE(Y)
6: **for** Label l in L **do**
7: $\quad C^l \leftarrow CD(X^l, K)$
8: $\quad O^l \leftarrow CR^w(X^l, C^l) \cup Y^l$
9: $\quad$ MATERIALIZE(O^l)
10: **end for**

Correctness Argument 2 (Algorithm 3). *X contains all the tuples that are present in any conflict from computing the conflicts on the whole input relation I. From Lemma 3.1, we know that $C \sqsupseteq C^l$ and thus no tuples from Y^l can end up in C^l. Therefore, we can safely use X^l as input to CD at each iteration and get the correct set of conflicts. By Theorem 3.3, we can add Y^l directly to the output in line 8. It now follows from the correctness argument of Algorithm 2 that this produces a correct output.* □

The CU algorithm isolates unconstrained tuples by copying them directly to the output. These tuples then do not take part in conflict derivation, reducing the input size to operations over $X^l \subseteq I^l$. As an additional observation, we note that the algorithm could be optimized for reuse if we could assert that the conflict set C does not change across any pair of labels. This may be particularly useful if one of the labels provides visibility over all tuples, allowing for reuse of the computation in line 1. For generality of presentation, we leave these data-dependent optimizations to future work, and focus on reasoning on the structure of conflict derivation functions and its relationship to data inclusions in security hierarchies.

Figure 8 shows us an illustration of how the evaluation of the first 3 lines of Algorithm 3 would look like on the example from Figure 4. As we can see, since t_0, t_1, and t_7 take part in no conflicts, they can be simply copied to the output respecting label visibility.

It is interesting to note that the CC and CU algorithms can be composed. While we omit details for brevity, we evaluate this method, named CU+, in our experiments (Section 4).

3.4 Strongly Monotonic *CD*, Disjoint Conflicts

In this section, we look at the special case where all conflicts generated by *CD* are disjoint. This special case appears in density constraints, which are important to reduce clutter in visualizations of spatial data [35]. Figure 9 depicts such a constraint on a small dataset of eight records with two labels: red and black, where red dominates black. We divide the visible area into a set of tiles, and all points occupying the same tile are in conflict with each other. For example, in tile A, t_1 and t_2 are in conflict. Now, suppose τ is one, such that in the output there is only one point per tile. Assume the surviving tuples are t_0, t_1, t_3, and t_5 for the red label. Notice that they are all black. Surely, if we could not access the red tuples, we would still pick the same four black tuples since they must have the highest weight in their respective tiles/conflicts. As such, there is an opportunity to reuse not only conflict derivation across security labels, but also conflict resolution.

To reason about reuse of conflict resolution, it is assumed that the maximum weight solution to a global selection is unique – when the conflicts are disjoint, there must be a unique selection of tuples in each conflict that yields the highest aggregated weight. In order to guarantee a unique solution, we assume that ties[2] are resolved by a unique identifier assigned to each tuple. For the remainder of this section, we assume CR^w to behave in this way.

Looking at Figure 9 again, the conflict from tile D only contains black points. Surely, the surviving point from tile D must be the same for the red and black labels, i.e., the winner between t_4 and t_5 is determined independently of tiles A, B, and C. Given a label l, we term a conflict *pure* for l if the labels of all tuples in the conflict are dominated by l. Otherwise, we call the conflict *impure*. Motivated the notion of pure and impure conflicts, we formalize a theorem operating on conflict solutions.

Theorem 3.4 (Reusing Conflict Solutions). *Let $A = (\{l_1, l_2, ..., l_n\}, \preceq) \in \mathbb{A}$ where $l_j \succeq l_i$ and CD be strongly monotonic producing disjoint conflicts. Given $R \in C^{l_j}$, if $\forall t \in R, t.label \preceq l_i$, then the output of resolving R subject to l_j can be reused for l_i.*

[2] A *tie* occurs when two or more tuples in the same conflict have the same weight.

Proof. Let R be any conflict from C^{l_j}. If $\forall t \in R, t.label \preceq l_i$, then R is also in C^{l_i} by Observation 1. Furthermore, given the assumption of disjoint conflicts, we know that no $t \in R$ can occur in another conflict. Therefore, R can be resolved independently from all other conflicts and the result of resolving R subject to l_j is equal to that of resolving R subject to l_i. □

Theorem 3.4 states that if a conflict is pure for a label, we can reuse the solution to said conflict for the label. For example, suppose we have computed the global selection for the red label (the output being t_0, t_1, t_3, and t_5). Now, in tile D, we can directly add t_5 to the output for the black label since the conflict of tile D is pure subject to the black label.

Algorithm 4 exploits the observation in Theorem 3.4 by reusing the result of CR^w for any pure conflicts for a given label.

Algorithm 4 : ***RS***
Reusing Conflict Solutions using Theorem 3.4

Require: User-defined set of constraints K
Require: Strongly monotonic conflict derivation function CD
Require: Conflict resolution function CR^w
Require: Input relation I
1: $C \leftarrow CD(I, K)$
2: $OC \leftarrow CR^w(I, C) \bowtie \Pi_{cid, i_1, i_2, ..., i_p}(C)$
3: MATERIALIZE(C)
4: MATERIALIZE(OC)
5: **for** Label l in L **do**
6: $\quad \delta^l \leftarrow \Pi_{cid}(\sigma_{label \succ l}(OC))$ // CIDs from impure conflicts
$\quad$ // Antijoin. Gives us the outputs we can reuse
7: $\quad OC^l \leftarrow OC \triangleright \delta^l$
$\quad$ // Unconstrained and tuples from impure conflicts
8: $\quad IX^l \leftarrow I^l - I^l \ltimes (C \triangleright \delta^l)$
9: $\quad O^l \leftarrow CR^w(IX^l, C^l) \cup \Pi_{i_1, i_2, ..., i_p}(OC^l)$
10: $\quad$ MATERIALIZE(O^l)
11: **end for**

Correctness Argument 3 (Algorithm 4). *In line 1, the conflicts are computed as in Algorithm 2, which is reused in line 9 of every iteration in accordance with Lemma 3.1. OC^l in line 7 gives us the tuples from O we can reuse by Theorem 3.4. That is, tuples in O that come from conflicts where the label of every tuple is dominated by l. That information is gained from δ^l where we store the ID of every conflict that has one or more tuples with a label dominating l. In line 8, we store the tuples that are not in any conflict or come from a conflict with an ID from δ^l. Thus, we use IX^l as input to CR^w in line 9 to resolve the* impure *conflicts. Now, $\Pi_{i_1, i_2, ..., i_p}(OC^l)$ contains the output of every* pure *conflict and $CR^w(IX^l, C^l)$ computes the output of every* impure *conflict, and thus the union is equal to O^l.* □

The RS algorithm decreases the input size to CR^w by identifying tuples that we can directly add to the output. However, we clearly observe that Algorithm 4 needs many operations to identify reusable tuples. Similarly to the CU+ algorithm, we can compose the reuse of conflict solutions with separation of constrained and unconstrained tuples. We omit details of this additional algorithm due to space constraints.

4 EXPERIMENTS

In this section, we evaluate experimentally the performance of the algorithms presented in this paper. The goals of the experiments are presented in Section 4.1 and their setup described in Section 4.2. Afterwards, we discuss the results obtained in Section 4.3.

4.1 Goals

The ultimate goal of the experiments is to to test the performance of the algorithms introduced in Section 3 compared to the naïve approach. An explicit non-goal is to evaluate the performance of the underlying global selection system for declarative cartography introduced by previous work [24, 25]. Since the global selection system needs to evaluate a set of user-defined constraints to derive visual conflicts and calculate a solution to an NP-hard conflict resolution problem, its purpose is to be run offline as a pre-computation step in advance of data visualization. This usage is in line with similar approaches [32, 35]. As calculating a global selection can be time-consuming, reducing the time necessary to run multiple global selections in connection with supporting visualizations with fine-grained access control is a worthy endeavor.

We show experiments that illustrate the performance of the algorithms proposed in this paper over a range of real datasets with different sizes as well as varied sizes and structures of security hierarchies. Specifically, the goals of our experiments are to demonstrate:

(a) How the algorithms scale on the number of labels in a linear security hierarchy (Section 4.3.1)
(b) How the complexity of the given constraints as well as the ratio between unconstrained and constrained tuples affect the performance of our algorithms (Section 4.3.2)
(c) How consistently the algorithms perform on DAG-shaped security hierarchies (Section 4.3.3)

We split the experimental results (Section 4.3) into subsections corresponding to the goals above.

4.2 Setup

4.2.1 Implementation. We build on a specialized version of the global selection implementation described in [24]. This implementation uses essentially the same algorithms for conflict derivation and resolution described in [25], but avoids as much as possible the use of temporary tables in the database for better performance. Conflict derivation is achieved by execution of the SQL of user-defined constraints, while conflict resolution employs a sorting-based greedy heuristic. As discussed in [25], this heuristic achieves a good trade-off in runtime and solution quality, and overall takes up a small percentage of the running time of a global selection compared to conflict derivation.

We argue that for the evaluation in this paper, the choice of this heuristic for CR^w is sufficient as follows. First, note that the proposed algorithms that potentially reduce the input to CR^w are CU, CU+, and RS. CU and CU+ only filter out unconstrained tuples. It is fair to assume that the runtime of any reasonable optimization procedure used in CR^w should be more strongly correlated with the size of the conflict set than with the number of unconstrained tuples (which should just be picked to be in the output if given to CR^w). So gains in performance from CU or CU+ should come overwhelmingly from savings in the computation of CD as opposed to CR^w. For RS, on the other hand, reductions in CR^w computation may be nontrivial. However, RS is only applicable when conflicts are disjoint. In this case, the sorting heuristic chosen is optimal, obviating the need to evaluate other optimization methods for CR^w.

We represent labels as attributes of input tuples. As mentioned earlier, these attributes are encoded in the pre-/post-plane when security hierarchies are trees [7], or alternatively with a prime number encoding when security hierarchies are general DAGs [43]. The overall skeleton of the algorithms in Section 3 includes a loop over all labels in the security hierarchy and a set of relational algebra expressions either before the loop or in the loop body. We leverage that structure in our implementation by taking a code generation approach. For each label in the security hierarchy and for any pre-loop code fragments, we generate pure SQL code in a global selection language preprocessor before executing the SQL code in PostgreSQL, the target database server. The algorithms introduced in this paper at selected points materialize information with the MATERIALIZE construct, which is implemented by creating temporary tables using the `UNLOGGED` keyword for performance.

4.2.2 Datasets. We use three different datasets that were part of the evaluation of [24]. Two of the datasets contain spatial data: Openflights Airports (7K points)[3] and OpenStreetMap points of interest (673K points).[4] The third dataset, Genius (13K rows),[5] contains information about lyrics from several rap artists. We create a spatial visualization from this dataset by 1-dimensional tiling of artists by the vocabulary sizes of their lyrics.

Henceforth, we call the datasets *airports*, *OpenStreetMap*, and *rappers*, respectively. We have chosen the datasets for two reasons. First, all datasets include real-world data, and are thus representative of skew to be found in practice. Second, the datasets offer a convenient frame of reference with respect to previous work [24, 25].

Since the original datasets do not include any access control information, we need to model a distribution of labels in the input relation in connection to a security hierarchy. For simplicity of interpretation, we focus on linear security hierarchies throughout the experiments, with exception of Section 4.3.3 where security hierarchies are generated through a random DAG generator. To assign labels to tuples in the input, we use two different distributions, *uniform* and *self-similar* [15]. With a uniform distribution, we assign each label with equal probability to a tuple. The self-similar method assigns the most dominated label randomly to 80% of the data. Then, the second most dominated label is assigned randomly to 80% of the remaining 20% of the data, and so on. Again for simplicity, we only use self-similar distributions on linear security hierarchies. They could, however, conceptually be applied to a tree or DAG, e.g., by uniformly assigning all labels from the lowest level to 80% of the data, and so on for each level in the hierarchy. Intuitively, the self-similar distribution models hierarchical structures in the real world, e.g., a company with only one CEO and many employees.

In Algorithms 2 to 4, we materialize information before we loop through the labels. Intuitively, the structure of the security hierarchy should therefore not matter as much as the number of labels.

[3] http://openflights.org/data.html
[4] http://planet.openstreetmap.org/
[5] http://rap.genius.com/

Moreover, we expect the distribution of labels to matter more than the hierarchy itself. For example, suppose a label is distributed such that the label has access to 1% of the input. Now, suppose we have another distribution were the label has access to 90% of the input. Intuitively, we expect the running time of the global selection for the latter case to be significantly higher than that of the former. In Section 4.3.3, we discuss whether or not this claim is supported by the experimental results.

4.2.3 Queries. The queries and constraints used for the global selections in our experiments are slightly modified versions of the ones presented in [24]. In summary, for both the airports and OpenStreetMap datasets, we use a density constraint that dictates that only 10 records may occupy the same tile on a map. The number of tiles on the map is controlled by the desired zoom level in a tile pyramid scheme [12]. If not otherwise stated, we use a zoom level of 4 corresponding to 256 tiles for all 2-D spatial data. As stated previously, we only use single-scale global selections in this work. For the airports dataset, we also test the proximity constraint, which we have used as a running example, with a distance of 10 pixels. For the rappers dataset, we use a constraint that maps the vocabulary (distinct words in lyrics) of a rapper to buckets of word-count intervals. We use an interval of 80 words. For brevity, we denote this constraint as the *wordcount* constraint. Note that the wordcount and density constraints are strongly monotonic, whereas the proximity constraint is only monotonic. However, the proximity constraint only generates conflicts of size two. Recall from Observation 2 that we therefore can use CC and CU+, despite the fact that these methods require strongly monotonic *CD*s.

We only expect CU and CU+ to improve performance when *some* tuples are unconstrained. Therefore, we refrain from showing results from these methods with the density and wordcount constraints, since in these cases *all* tuples are constrained by definition.

4.2.4 System Configuration. The machine used for all experiments is a Lenovo T460s laptop with a dual-core 2.4 GHz Intel Core i5-6300U processor, 12GB of RAM, and a 256 GB NVMe Samsung MZ-VPV256 SSD. The system runs Microsoft Windows 10 Enterprise, and we employ a PostgreSQL 9.5 server with best-effort tuning of the default settings. An important setting for the PostgreSQL server is to disable `autovacuum`. In PostgreSQL, vacuum is an operation to garbage-collect a database.[6] To avoid interference from the `autovacuum` service on the runtime, we manually vacuum the entire database between runs. The other settings tuned allow the database server to use more main memory for different tasks to speed up the experiment process. In particular, we set shared buffers to 2GB, work memory for sorting and hashing to 0.5 GB, maintenance work memory for vacuuming to roughly 1.5GB, and turn `fsync` off.

4.2.5 Experimental Methods. Every experiment follows a general pattern with given specifications. The general pattern can be summarized in 8 steps: (1) Vacuum database. (2) Generate security hierarchy. (3) Generate label encodings using specified encoding method. (4) Assign labels to tuples in input table according to a given distribution. (5) Begin measuring time. (6) Compute the global selection using the specified algorithm. (7) End measuring time. (8) Delete all generated tables. The specifications determine among others: the algorithm to use, the input table to the global selection operator, the security hierarchy, and the distribution of labels. For every configuration, we run the general pattern 10 times and calculate the average runtime of step 6 as well as the standard deviation. We observed very consistent running times for all algorithms.

[6] https://www.postgresql.org/docs/9.5/static/sql-vacuum.html

4.3 Experimental Results

4.3.1 Scalability on Number of Labels. In this section, we report on the scalability of our algorithms on the number of labels in a linear security hierarchy. Figures 10 to 15 report the performance of each algorithm on all datasets with self-similar and uniform distributions, respectively. The number of labels ranges from 1 to 64. With only one label in the security hierarchy, we do not expect any performance gain from any of the algorithms. On the contrary, we can observe the overhead introduced by each approach when compared with Naive. CC introduces a minute overhead in all the experiments – evidencing the comparatively low cost of materializing conflicts once they are derived. CU, by contrast, has higher overhead. For example, in Figure 13, CU takes roughly 2x the running time of Naive. Recall from Section 3.3 that the implementation of CU computes the conflicts twice for a single label. Therefore, this large overhead is not surprising, given that the algorithm has no opportunity to reuse results for unconstrained tuples over other labels. Since CU+ does *not* compute the conflicts twice, CU+ has a slightly smaller overhead than CU. The overhead of RS is generally bigger than CC, but smaller than CU.

Throughout Figures 10 to 15, we observe that CC dominates Naive whenever the number of labels is bigger than one. In addition, the relative performance gap between CC and Naive increases as the number of labels increases, reaching a factor of roughly 2x-5x. CU and CU+ are only evaluated with the proximity constraint, since only this constraint allows for unconstrained tuples. In Figures 12 and 13, 30% of the tuples are unconstrained; still, CU's overhead in identifying those tuples makes it perform worse than Naive in most configurations. Even though CU+ performs substantially better than Naive, it performs slightly worse than only CC.

In Figures 10 and 11, we observe that RS outperforms Naive as the number of labels increases. We also see that RS does so earlier when the distribution is uniform than when the distribution is self-similar. This effect is surprising in that using self-similar distributions, we would expect the probability of a conflict being pure to be higher since the least dominating labels are represented more in the data. So the effect suggests that Algorithm 4 incurs too high costs in identifying conflicts as pure. A similar reasoning can be applied to Figures 14 and 15, except that the costs of RS with the OpenStreetMap dataset are higher and the method performs more similarly to Naive. We conjecture that this latter behavior comes from the fact that the OpenStreetMap dataset is substantially larger than the rappers dataset — since we map many more points to a single tile, we would also expect the probability of pure conflicts to be smaller. In summary, RS ends up being dominated by CC, which emerges as the best method. We also ran the density constraint on the airports dataset which showed similar results.

In Figures 12 and 13, we observe that CC performs much better relative to Naive compared to the experiments using density constraints. Moreover, CC scales significantly better on the number of

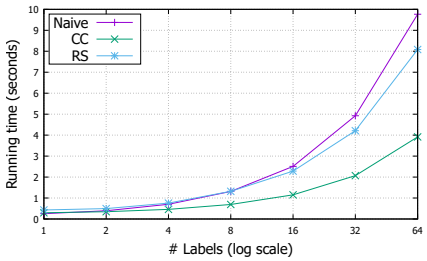

Figure 10: Rappers with wordcount constraint and self-similar distribution.

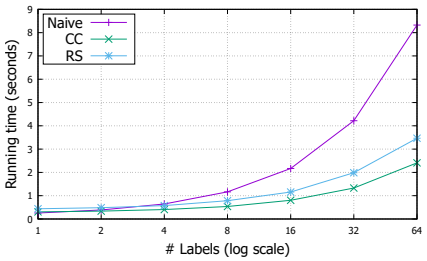

Figure 11: Rappers with wordcount constraint and uniform distribution.

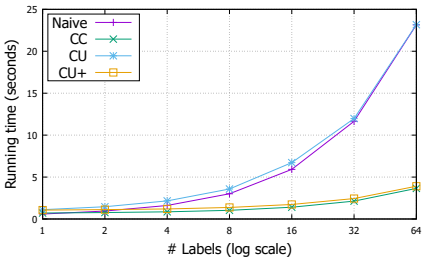

Figure 12: Airports with proximity constraint and self-similar distribution.

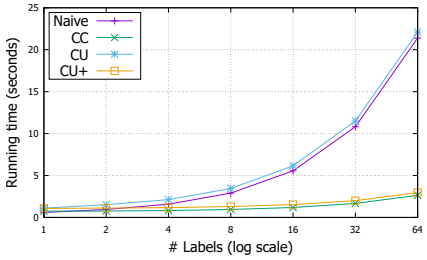

Figure 13: Airports with proximity constraint and uniform distribution.

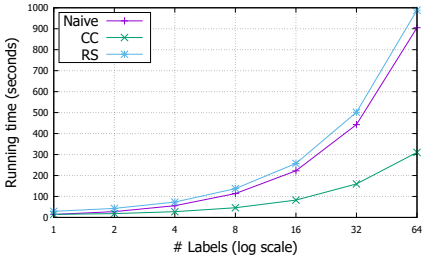

Figure 14: OpenStreetMap with density constraint and self-similar distribution.

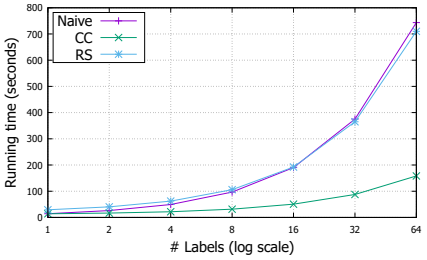

Figure 15: OpenStreetMap with density constraint and uniform distribution.

labels with the proximity constraint. Computing the conflicts for the proximity constraint is much more expensive than for the density constraint since the proximity constraint requires a spatial join. Our results confirm that the benefit of reusing conflicts increases the more expensive conflict derivation becomes.

Finally, we observe that Naive, CC, CU, and CU+ overall have a longer running time using the self-similar distribution of labels. If we accumulate the sizes of the input relations subject to each label for both distributions, the self-similar distribution results in a much higher total number of tuples to be processed by global selections. For example, in a four-label linear hierarchy, the cumulative size of the self-similar distribution is 375.2% compared to 250% of the input relation for the uniform distribution.

4.3.2 Conflict Complexity. We have hinted before at the notion of division of labor between CR^w and CD. In this section, we aim to explore how the ratio between constrained and unconstrained tuples as well as the complexity of the constraints affect the performance of the algorithms. The problem with the density constraint as well as the wordcount constraint is that every tuple is always in exactly one conflict. The proximity constraint is the only one used where not all tuples are necessarily in a conflict. With this constraint, we can actually influence the ratio between constrained and unconstrained tuples through the zoom level at which we compute the global selection. A zoom level of 1 is where we can see the whole world (e.g., Figure 1), and as the zoom level increases, we move closer to the surface. As such, we would expect more tuples to become unconstrained as the zoom level increases.

Figure 16 shows how the various methods behave with the proximity constraint on the airport dataset using a self-similar distribution of labels. We also ran the experiment using uniform distribution which yielded similar results. Unlike the previous experiments where we varied the number of labels, we here vary the zoom level. We use a linear security hierarchy with eight labels. At zoom level 1, 99% of all tuples are in at least one conflict. At zoom level 4, which we have used for all the previous experiments on the airports and OpenStreetMap datasets, 70% of all tuples are constrained. At zoom level 7, only 12% are in a conflict. We observe that CU performs very poorly compared to Naive when most tuples are constrained. CU performs better relative to Naive as the zoom level (and thus the percentage of unconstrained tuples) increases, although always slightly worse. This behavior is similar to what was observed in Section 4.3.1. Likewise, CU+ is no better than CC.

4.3.3 DAG-Shaped Security Hierarchies. In this section, we explore how the algorithms perform on security hierarchies with more complex structure than linear. For this experiment, we fix the number of labels at 10 and randomize the structure of the security hierarchy using a random DAG generator.[7] The generated DAG always has a single root node, and the depth of the DAG varies between three and five. Recall from Section 4.2 that we employ a uniform distribution with DAG security hierarchies when assigning labels to input tuples. As for the previous experiments, we ran this experiment ten times on each configuration and randomized the DAG for each run.

Figures 17 and 18 report the results for the airports and OpenStreetMap datasets. We observe that the standard deviation is very small across the board, which supports our claim from Section 4.2 that the number of labels and distribution hereof play a bigger role on the running time than the structure of the security hierarchy. The relative running times between the algorithms show the same trends we have seen in the previous experiments. We also ran this experiment on the rappers dataset with the wordcount constraint and on airports with the density constraint, obtaining similar relative performance between Naive, CC, and RS as in Figure 18.

[7] http://www.graphdrawing.org/data.html

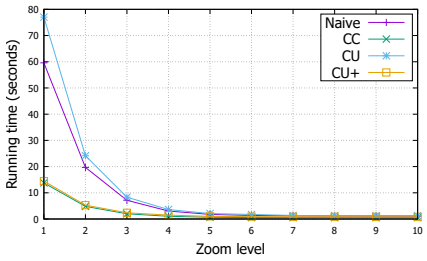

Figure 16: Airports with proximity constraint and self-similar distribution.

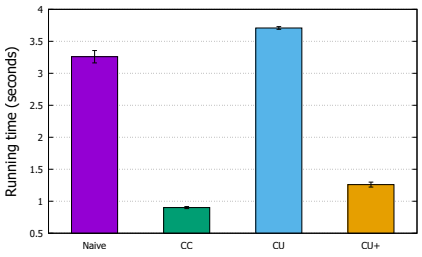

Figure 17: Airports with proximity constraint on random DAG hierarchies.

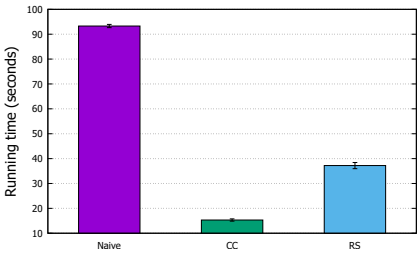

Figure 18: OpenStreetMap with density constraint on random DAG hierarchies.

5 RELATED WORK

Several proposals introduce access control models based on geographical context, e.g., a certain user is only authorized to see data within a region of a city [3, 5, 11, 30, 36, 41]. Furthermore, in some of these proposals, e.g., GEO-RBAC [6], SRBAC [18], LoT-RBAC [8], or the method of Shebaro et al. [37], role activation may depend on the location of the user and roles can thus change dynamically as users move. Moreover, role hierarchies can in some cases be defined [3, 5, 6, 8, 18]. In contrast to our approach, however, little is said about how to efficiently reuse computation of access control permissions across hierarchical levels over such geospatial data or how to integrate these models with map generalization. In addition, our model is based on label-based access control, which fits more naturally use cases such as agricultural extension services than role-based access control (see Section 6).

Maps are the visualization category of choice for geospatial data. Many data management systems for visualizations have been proposed over time [2, 26, 29], some of which include maps [4, 17]. However, none of these visualization systems have considered the issue of optimizing data reduction operations in the presence of access control. Furthermore, in data management systems for visualizations, the two primary data reduction approaches employed are data selection (e.g., by filtering or random sampling) and data aggregation. Our work focuses on data selection, since the problem of data aggregation under access control can often be substantially simpler to solve. Consider, for example, the data aggregation approach of Nanocubes [27], in which a multidimensional aggregate representation inspired by the Dwarf index [39] is linked to the nodes of a quadtree. To adapt the approach to access control, we can conceptually replace each aggregate in the data structure by a list of aggregates per access label. If a simple linear representation of such a list proves to be too space-consuming, we could further compress the representation of the aggregates per label by exploiting the security hierarchy structure in a delta encoding scheme.

In contrast to data aggregation, deriving data selections subject to access control can be far more challenging and computationally expensive. In particular, data selection methods have been recently proposed that respect user-centered visualization constraints and/or optimization goals [16, 24, 25, 32, 35]. These methods are advantageous in that they create visualizations that are more representative to end users when compared with random sampling, especially over multiple scales in cartographic use cases [16, 32, 35]. However, these methods often involve computing solutions to hard optimization problems. Given their degree of algorithmic sophistication, it is non-trivial to reuse computations in these methods when deriving multiple data selections for a security hierarchy. This paper shows that a naïve combination of access control with an optimization-based method of data selection for visualizations yields runtimes that grow quickly with the size of security hierarchies. To address this issue, the paper presents a set of new theoretical results and algorithms. A novel aspect of the algorithms proposed in this paper is the observation that the data inclusion structure originating from a label-based access control scheme can be leveraged towards reusing derivations of visual conflicts across different access roles in a data selection method for declarative cartography.

6 CASE DISCUSSION

This paper is motivated by problems encountered while working on the Future Cropping project in Denmark.[8] Future Cropping is a collaboration between, among others, the farming information provider SEGES,[9] companies, and universities aiming to improve farming using modern technology. Agriculture takes up 62% of the area of Denmark[10] and is responsible for 25% of the value of goods exported by the country.[11] Unsurprisingly, there is substantial interest in improving farming practices.

Agricultural extension is a key element to achieving such improvements. Extension services allow farm consultants to provide customized advice to farmers. In Denmark, SEGES is the main information provider for extension services, offering a platform used by both farm consultants and farmers. The SEGES platform includes several web services with data on fields, satellite imagery, crops, among many others.[12] In Future Cropping, SEGES's data platform is being extended further with new analytical services combining multiple data sets, e.g., to improve fertilizer application by exploiting satellite imagery or to support decision making by exploiting data collected from farm machinery.[13]

In applications such as the SEGES platform, farmers own their data, but can give access to consultants. Consultants can access the data from multiple farmers, compare data, and guide each farmer on how to proceed to maximize her yield. With increasing data density coming from farm equipment and other sensors, the need for data reduction methods in creating maps under access control is expected to grow.

[8] https://futurecropping.dk/
[9] https://www.seges.dk
[10] https://www.dst.dk/Site/Dst/Udgivelser/nyt/GetPdf.aspx?cid=24323
[11] https://www.lf.dk/om-os/vores-bidrag/25-pct-af-danmarks-eksport
[12] https://www.seges.dk/software/plante/mark-online
[13] https://futurecropping.dk/guldet-ligger-gemt-i-de-rigtige-data/

In this agricultural extension scenario, a natural security hierarchy contains labels for farmers, consultants, and system administrators. Moreover, data items are labeled according to the farmer they belong to. For example, in a relation with field polygons and associated attributes, tuples are labeled according to which farmer owns each field. In other words, since schemas are shared among farmers, we employ fine-grained access control to ensure farmer data is only accessed by authorized users. These authorizations are managed by the information provider, e.g., SEGES, who can alter the security hierarchy. For example, a given consultant label should dominate the farmer labels corresponding to the data the consultant is allowed to see. A system administrator label can dominate all farmer labels in the hierarchy such that the correspoding farmers have authorized SEGES to use their data for running data mining algorithms according to the terms of service.

7 CONCLUSION

In this paper, we study the problem of efficiently computing global selections for declarative cartography under fine-grained access control. The key idea of our approach is to increase reuse in the computation of global selections across access labels. Our theoretical results lay the groundwork for algorithms exploring various reuse approaches. All algorithms generate the same output as the naïve approach of computing the global selection from scratch for each label in a security hierarchy, thus guaranteeing that access control is not violated. Our experimental results show that reusing conflicts between global selections across multiple access roles improves runtime significantly compared to the naïve approach. This method dominates the naïve approach for any security hierarchy of at least two access labels and the benefit increases with the number of labels as well as with the complexity of the constraints.

REFERENCES

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of Databases.* Addison-Wesley.

[2] A. Aiken, J. Chen, Michael Stonebraker, and A. Woodruff. 1996. Tioga-2: a direct manipulation database visualization environment. In *Proc. ICDE*. 208–217.

[3] Vijayalakshmi Atluri and Soon Ae Chun. 2004. An Authorization Model for Geospatial Data. *IEEE Trans. Dependable Secur. Comput.* 1, 4 (2004), 238–254.

[4] Leilani Battle, Michael Stonebraker, and Remco Chang. 2013. Dynamic reduction of query result sets for interactive visualizaton. In *Proc. IEEE Big Data*. 1–8.

[5] A. Belussi, E. Bertino, B. Catania, M. L. Damiani, and A. Nucita. 2004. An Authorization Model for Geographical Maps. In *Proc. GIS*. 82–91.

[6] Elisa Bertino, Barbara Catania, Maria Luisa Damiani, and Paolo Perlasca. 2005. GEO-RBAC: A Spatially Aware RBAC. In *Proc. SACMAT*. 29–37.

[7] Peter A. Boncz, Stefan Manegold, and Jan Rittinger. 2005. Updating the Pre/Post Plane in MonetDB/XQuery. In *Proc. <XIME-P/> Workshop*.

[8] Suroop Mohan Chandran and James B. D. Joshi. 2005. *LoT-RBAC*: A Location and Time-Based RBAC Model. In *Proc. WISE*. 361–375.

[9] Andrew Crotty, Alex Galakatos, Emanuel Zgraggen, Carsten Binnig, and Tim Kraska. 2016. The Case for Interactive Data Exploration Accelerators (IDEAs). In *Proc. HILDA Workshop*. 11:1–11:6.

[10] Andrew Curtis, Jacqueline W. Mills, and Michael Leitner. 2006. Keeping an eye on privacy issues with geospatial data. *Nature* 441 (05 2006), 150 EP –.

[11] Maria Luisa Damiani and Elisa Bertino. 2007. Access Control Systems for Geospatial Data and Applications. In *Belussi A., Catania B., Clementini E., Ferrari E. (eds) Spatial Data on the Web*. 189–214.

[12] L. De Cola and N. Montagne. 1993. The pyramid system for multiscale raster analysis. *Computers and Geosciences* 19 (1993), 1393–1404.

[13] Future Cropping [n. d.]. Future Cropping partnership website. ([n. d.]). Available at https://futurecropping.dk/en/, last accessed September 8, 2017.

[14] Michael F. Goodchild. 2010. Twenty years of progress: GIScience in 2010. *J. Spatial Information Science* 1, 1 (2010), 3–20.

[15] Jim Gray, Prakash Sundaresan, Sussanne Englert, Ken Baclawski, and Peter J. Weinberger. 1994. Quickly generating billion-record synthetic databases. In *Proc. ACM SIGMOD*. 243–252.

[16] Tao Guo, Kaiyu Feng, Gao Cong, and Zhifeng Bao. 2018. Efficient Selection of Geospatial Data on Maps for Interactive and Visualized Exploration. In *Proc. ACM SIGMOD*.

[17] Pat Hanrahan. 2012. Analytic Database Technologies for a New Kind of User: The Data Enthusiast. In *Proc. ACM SIGMOD*. 577–578.

[18] Frode Hansen and Vladimir Oleshchuk. 2003. SRBAC: a spatial role-based access control model for mobile systems. In *Proceedings of the 7th Nordic Workshop on Secure IT Systems, Norway*.

[19] Lars Harrie and Robert Weibel. 2007. Modelling the overall process of generalisation. *Generalisation of geographic information: cartographic modelling and applications* (2007), 67–88.

[20] Ponemon Institute. 2017. 2017 Cost of Data Breach Study, Global Overview. (2017). Sponsored by IBM Security; available at http://www-03.ibm.com/security/data-breach/, last accessed September 8, 2017.

[21] Dean F. Jerding and John T. Stasko. 1998. The Information Mural: A Technique for Displaying and Navigating Large Information Spaces. *IEEE Transactions on Visualization and Computer Graphics* 4, 3 (1998), 257–271.

[22] Uwe Jugel, Zbigniew Jerzak, Gregor Hackenbroich, and Volker Markl. 2014. M4: A Visualization-Oriented Time Series Data Aggregation. *PVLDB* 7, 10 (2014), 797–808.

[23] Govind Kabra, Ravishankar Ramamurthy, and S. Sudarshan. 2007. Fine Grained Authorization Through Predicated Grants. In *Proc. ICDE*. 1174–1183.

[24] P.K. Kefaloukos. 2015. *Database-Integrated Multi-Scale Selection for Map and Data Visualizations.* Ph.D. Dissertation. Department of Computer Science, University of Copenhagen (DIKU).

[25] P.K. Kefaloukos, Marcos Vaz Salles, and Martin Zachariasen. 2014. Declarative Cartography: In-Database Map Generalization of Geospatial Datasets. In *Proc. ICDE*. 1024–1035.

[26] R. Krishnamurthy and M. Zloof. 1995. RBE: Rendering by example. In *Proc. ICDE*. 288–297.

[27] Lauro Didier Lins, James T. Klosowski, and Carlos Eduardo Scheidegger. 2013. Nanocubes for Real-Time Exploration of Spatiotemporal Datasets. *IEEE Trans. Vis. Comput. Graph.* 19, 12 (2013), 2456–2465.

[28] Zhicheng Liu, Biye Jiang, and Jeffrey Heer. 2013. *imMens*: Real-time Visual Querying of Big Data. *Comput. Graph. Forum (Proc. EuroVis)* 32, 3 (2013), 421–430.

[29] M. Livny, R. Ramakrishnan, K. Beyer, G. Chen, D. Donjerkovic, S. Lawande, J. Myllymaki, and K. Wenger. 1997. DEVise: Integrated Querying and Visual Exploration of Large Datasets. In *Proc. ACM SIGMOD*. 301–312.

[30] Andreas Matheus. 2005. Declaration and Enforcement of Fine-grained Access Restrictions for a Service-based Geospatial Data Infrastructure. In *Proc. SACMAT*. 21–28.

[31] Anna Monreale, Gennady Andrienko, Natalia Andrienko, Fosca Giannotti, Dino Pedreschi, Salvatore Rinzivillo, and Stefan Wrobel. 2010. Movement Data Anonymity Through Generalization. *Trans. Data Privacy* 3, 2 (2010), 91–121.

[32] Yongjoo Park, Michael J. Cafarella, and Barzan Mozafari. 2016. Visualization-aware sampling for very large databases. In *Proc. ICDE*. 755–766.

[33] Shariq Rizvi, Alberto Mendelzon, S. Sudarshan, and Prasan Roy. 2004. Extending Query Rewriting Techniques for Fine-Grained Access Control. In *Proc. ACM SIGMOD*. 551–562.

[34] Walid Rjaibi and Paul Bird. 2004. A Multi-Purpose Implementation of Mandatory Access Control in Relational Database Management Systems. In *Proc. VLDB*. 1010–1020.

[35] Anish Das Sarma, Hongrae Lee, Hector Gonzalez, Jayant Madhavan, and Alon Halevy. 2013. Consistent Thinning of Large Geographical Data for Map Visualization. *ACM Trans. Database Syst.* 38, 4 (2013), 22:1–22:35.

[36] Liliana Kasumi Sasaoka and Claudia Bauzer Medeiros. 2006. Access Control in Geographic Databases. In *Advances in Conceptual Modeling - Theory and Practice, ER 2006 Workshops (CoMoGIS)*. 110–119.

[37] Bilal Shebaro, Oyindamola Oluwatimi, and Elisa Bertino. 2015. Context-Based Access Control Systems for Mobile Devices. *IEEE Trans. Dependable Sec. Comput.* 12, 2 (2015), 150–163.

[38] Shashi Shekhar, Steven K. Feiner, and Walid G. Aref. 2015. Spatial Computing. *Commun. ACM* 59, 1 (2015), 72–81.

[39] Yannis Sismanis, Antonios Deligiannakis, Nick Roussopoulos, and Yannis Kotidis. 2002. Dwarf: shrinking the PetaCube. In *Proc. ACM SIGMOD*. 464–475.

[40] CACM Staff. 2017. The Internet of Things. *Commun. ACM* 60, 5 (2017), 18–19.

[41] Ebrahim Tarameshloo and Philip W.L. Fong. 2014. Access Control Models for Geo-social Computing Systems. In *Proc. SACMAT*. 115–126.

[42] Qihua Wang, Ting Yu, Ninghui Li, Jorge Lobo, Elisa Bertino, Keith Irwin, and Ji-Won Byun. 2007. On the Correctness Criteria of Fine-Grained Access Control in Relational Databases. In *Proc. VLDB*. 555–566.

[43] Gang Wu, Kuo Zhang, Can Liu, and Juan-Zi Li. 2006. Adapting Prime Number Labeling Scheme for Directed Acyclic Graphs. In *Proc. DASFAA*. 787–796.