Working document for revision of ICDE '14 paper

Kostas, Martin, Marcos

November 19, 2013

1 About

Notes and overall plan recorded in Kostas/Marcos Skype chat October 16 2013. Citations for ladder/star approach in Kostas/Pia chat November 12 2013. Citations for NP-hardness in Kostas/Martin email.

2 Renamings and structural changes, to be consistent with literature

- Perhaps rename "filtering problem" to "thinning problem" or "selection problem"; consistent with the *selection operator* in generalization literature, or *thinning* in the fusion tables paper. Selection problem is most consistent with the literature.
- Rename "bottom-up" to "ladder" approach because it is more consistent with the literature [3]
-

3 Why map conflict resolution to an NP-hard problem?

Constraint based modeling is NP-hard (tags: algorithm section) If we model the problem using conflict sets, the problem is definitely NP-hard for all interesting cases.

For an informal proof, consider the vertex cover problem. Given a graph G=(V,E), find a minimum size subset of the vertices S such that every edge in E has an endpoint in S. Think of the vertices as records and the edges as conflict sets. This problem would then model the filtering problem for the special case where all conflict sets had exactly two records (and all records had identical weights).

The vertex cover problem is NP-hard, even for very constrained cases. For example, even if G is a planar graph and every vertex has degree at most 3, the problem remains NP-hard [4]. In other words, even if the conflict sets only have two records each, and each record is involved in at most 3 conflict sets, the problem remains NP-hard. It is hard to imagine that any interesting application is more restrictive.

Note that the cubic vertex cover problem is APX-hard - which means that it cannot be approximated arbitrarily close unless P=NP [1]. So even this restricted version is "really" hard.

4 What we do and why

What is generalization bla bla bla

Why constraints? One approach to modeling the generalization process is *constraint based modeling* [5]. In this approach, constraints are used to discriminate between good and bad solutions. The formulation of algorithms for computing solutions is considered a secondary issue.

It is usually described as the process of finding a compromise between conflicting constraints of legibility and representation, with a cost associated with violating constraints. In this formulation the solution space is typically infeasible, and the objective is to find the infeasible solution with the smallest associated cost [6].

instead of methods for computing solutions [2]. This is essentially a declarative way to generalize spatial data, so it is a good starting point for "declarative cartography".

Why implicit objective function? Defining the search space of acceptable solutions is not enough. There should also be a way to discriminate between good and bad solutions. In the literature, constraint based modeling

(bum bum)

There are other ways to formulate the problem. If the constraints used are non-conflicting, e.g. there are only legibility constraints, then one way to measure solutions is to use an objective function. This is the approach we take. In our approach, users do not formulate objective functions directly. Instead, users decide the record weights that make up the core of an implicit objective function. We argue that this is easier to understand for non-technical users, instead of having to declare a complex objective function explicitely.

Why combinatorial optimization? In this work we resolve conflicts by deleting a subset of the records in the dataset, known as *selection* in the generalization literature [8]. Combinatorial optimization is a particularly good fit for this type of generalization, because for each object there are only two possible states; deleted or not deleted. This means that the discrete search space is small when compared to the search space of e.g. the displacement operator [6].

Why only selection? There are many other operators, besides selection, for generalizing spatial data. However, the selection operator is arguably most appropriate for high volume point datasets, such as geocoded images and social media messages. When dataset have millions or billions of records, there is not much choice but to omit some of the records, and the selection operator is a good fit. Alternatively, aggregation is a possibility, but aggregating photos and messages is not immediately meaningful.

Justify choice of constraints In the context of generalization, constraints can be divided into *legibility constraints* and *representation constraints* [6]. Legibility constraints serve the purpose of ensuring that a map is *readable*. The purpose of representation constraints is to ensure that various complex properties of the input data are retained. Any constraint must have a measure in order to be operational, but the measures needed to express representation constraints are typically very complex. Examples of complex measures include spatial distribution and topological relationships in the data. In contrast to this, legibility constraints can usually be expressed in terms of scalar measures such as distance and counts. This makes them easier to understand and parameterize for users and easier to implement. For this reason, we have postponed support for representation constraints in our implementation, and focused entirely on legibility constraints.

Justify objective function Leaving out the representation constraints, means that it is trivial to satisfy the legibility constraints. Deleting all the records produces an empty dataset, which does not violate any expressible legibility constraint. To counter this, we formulate an objective function, which serves the purpose of retaining as many records in the dataset subject to the legibility constraints. Instead of treating all records equally, we allow users to assign a weight to the records, that intuitively indicates the importance of individual records. The assignment of weights is done statically. To counter the effect of the legibility constraints, the objective is to minimize the combined weight of records that are removed from the dataset. This can potentially lead to the situation were a record with high weight is removed to make room for two records of lesser weight. To ensure that this does not happen, is the responsibility of the algorithm used to compute a solution. In fact, many greedy algorithms will ensure that it does not happen, while compromising global optimality.

Justify combinatorial optimization Constraints serve the purpose of defining feasible solutions in the search space, and must be complemented by an effective method for finding a good solution. To compute a solution, we use a combinatorial optimization approach. Because or our limitations, records can have only two states. This makes combinatorial optimization particularly appropriate.

Be exploiting a regularity in the legibility constraints, we are able to map to a well-known optimization problem, the Set Multicover Problem. The advantage of mapping to a well-known problem is that we can reuse existing algorithms. While the Set

Multicover Problem is NP-hard, it is not harder than the generalization problem as we have stated it, since it is also NP-hard.

Multi-scale generalization We generalizing datasets for multiple scales at once, which is a typical requirement for zoomable maps found online. To generalize datasets for multiple scales, two overall approaches have been covered in the literature: the *ladder* approach and the *star* approach [3]. In the ladder approach, generalized datasets at lower scales are recursively derived from datasets generalized for larger scales. This means that the zoom-consistency constraint is automatically enforced. Enforces the zoom-constraint is appropriate for most, but not all, types of datasets. For this reason we use the ladder approach. Specifically, for generalization of regional labels at multiple scales, the star approach is arguably a better fit. This is because it allows labels to appear only on intermediate zoom-levels. Adding support for the star approach is a low hanging fruit for future work.

5 Why use conflict sets to model problem?

In this work we take a declarative approach to the generalization of spatial datasets. In this approach, the user should state only what is desired of an ideal result, without stating the means of producing the result. A good fit for this requirement is *constraint based modeling* [2, 5]. An advantage of this approach is that the definition of acceptable solutions is decoupled from the algorithms that compute solutions. This is an ideal starting point for a declarative approach to generalization.

HERE!!! remember to include equal shares of what and why.

Constraints implicitly define a search space of acc

Various typologies of constraints have been discussed in the literature [2, 6]. In the typology of Harrie and Weibel [6], there are two main categories of constraints; *legibility constraints* and *representation constraints*. Of these two, we only consider *legibility constraints* such as proximity and density constraints, that require a map to be readably. We do not handle *appearance constraints* that serve the purpose of maintaining faithful representation of the input data, such as maintaining the *spatial distribution* and *topological relationships* in a dataset. Handling representation constraints is important, and should be the focus of future work.

Legibility constraints and representation constraints are inherently at odds, which can be exploited to find a well balanced solution.

. Deleting all the records produces a very legible but also non-informative result. Not modifying

Regardless of which particular algorithm is used to generalize the data, one can define a subset of operations that can be performed on the data, such as deleting records, combining records and modifying records. In this work we only consider

Several methods have been proposed for finding good or optimal solutions in this search space, including agent based modeling, combinatorial optimization and continuous optimization [6]. In this work we use combinatorial optimization.

The core idea is to let the conditions that should be fullfilled by the generalized dataset act as constraints .

, were conditions can be grouped into legibility and representation [6].

A good generalization of a dataset should satisfy several *conditions*. We use a *constraint based modeling* approach for generalizing a spatial dataset [5]. Using constraints acceptable solutions creates a search space of acceptable solutions

without being explicit about the actions needed to produce solutions, leaves flexibility in choosing the best algorithms for the job [2].

A good generalization of a dataset should satisfy several *conditions*. The main idea behind *constraint based modeling* is to let these conditions act as constraints in the generalization process.

A big advantage of using a constraints is that they are not bound to a certain action [2], but instead define a search space of feasible solutions. The basic idea is to combine constraints with an approach for finding a good solution in the space of feasible solutions [6]. One such approach is combinatorial optimization, which is the approach

We approach the generalization of a dataset as a *constraint satisfaction problem*, which is a well known approach in the literature on generalization [2, 5]. In the typology of Harrie and Weibel [6], we consider only *legibility constraints* such as proximity and density constraints, that serve to purpose of making a map readable. We do not yet handle *appearance constraints* that serve the purpose of maintaining faithful representation of the input data, such as maintaining the *spatial distribution* and *topological relationships* of a dataset. Handling appearance constraints is future work.

The advantage of using a constraint based modeling approach, instead of e.g. a *condition-action* approach [6], is that it leaves flexibility in defining the method by which a satisfaction of the constraints is sought [2]. Being agnostic of the algorithms used to satisfy constraints is particularly good fit for designing a declarative language such as cvl. A constraint based approach also offers the opportunity to change the implementation as new algorithms for generalization are developed.

6 Refinements based on knowledge gained in Zürich

Why we use constraint based approach The advantage of using a constraint based modeling approach, instead of e.g. a *condition-action* approach [6], is that it leaves flexibility in defining the method by which a satisfaction of the constraints is sought [2]. Being agnostic of the algorithms used to satisfy constraints is particularly good fit for designing a declarative language such as cvl. A constraint based approach also offers the opportunity to change the implementation as new algorithms for generalization are developed.

Why we use combinatorial optimization In this work we resolve conflicts by deleting a subset of the records in the dataset, known as *selection* in the generalization literature [8]. Combinatorial optimization is a particularly good fit for this type of generalization, because for each object there are only two possible states; deleted or not deleted. This means that the discrete search space is small when compared to the search space of e.g. the displacement operator [6].

Requirements of a generalized map There are two main requirements that a generalized map must satisfy. Firstly, it must be a correct, though abstracted, representation of the physical reality. Secondly, the map must be readable by the user. If both these two requirements are not taken into account the map will not be of high quality. The final generalized map will eventually be a compromise between good readability and good representation [6].

Constraint based modelling (tags: method, related work) (From "Modelling the overall process of generalisation" [6] by Harrie and Weibel)

A generalized map should satisfy several *conditions*. The main idea behind *constraint based modeling* is to let these conditions act as *constraints* in the generalization process. The advantage of using constraints is that they are not bound to a certain action [2]. In other words, the definition of constraints is decoupled from the methods that resolve conflicts.

To use constraints in map generalization they must have a *measure*. In order for the generalized map to satisfy all constraints all the measures must have satisfying values, e.g. the distance between objects must be greater than some minimum threshold.

Most often, however, there exists no generalization solution that completely satisfies the constraints and the solution must be a compromise. To find the best compromise there must be a *cost* connected to the violations of the constraints (computed by the measure). The optimal generalized map is then the solution with the lowest total cost. Several methods for finding the optimal solution have been described in the literature, including *agent modelling*, *combinatorial optimization* and *continuous optimization* [6].

Note that one should be careful with the use of the word *optimal* here. That a generalized map is an optimal solution, in a constraint based modeling sense, does not mean anything more than that the constraints used are violated as little as possible. This implies that a solution will never be better than the constraints used. To compute a perfect generalized map would in addition require that we have a complete set of constraints for the generalized map.

Currently, the constraints used for generalization are limited in the sense that they cannot describe all aspects of a generalized map; hence, the generalized map using constraint based modeling is not optimal in a more *general* sense.

Typology of constraints (tags: limitations, related work) Various typologies of constraints have been discussed in the literature [2, 6]. In the typology of Harrie and Weibel [6], we consider only *legibility constraints* such as proximity and density constraints, that require a map to be readably. We do not handle *appearance constraints* that serve the purpose of maintaining faithful representation of the input data, such as maintaining the *spatial distribution* and *topological relationships* of a dataset.

Our current approach is to use an *objective function* to balance the legibility constraints, such that the combined weight of objects that are removed is minimized.

(Note: Stephan Schmid wrote a nice master's thesis that includes a thorough discussion of constraints [7]. Ask Weibel about what grade it got).

Future work on constraints in general The main limitation of the constraint based techniques today are the limitations of the constraints themselves. Important cartographic constraints still remain to be defined, others are poorly defined. More work is, for instance, required in formulating constraints, and associated measures, for groups of objects. Work on pattern recognition and on constraints for preserving cartographic patterns will be particularly important in this context [6].

Ladder/star approach (tags: method, limitations, related work) When generalizing a dataset for several scales, an important distinction is whether to follow the *ladder* or *star* approach [3]. We have only implemented the ladder approach, where lower scales are recursively derived from higher scales. An advantage of using the ladder approach is that the zoom-consistency constraint is automatically satisfied.

While enforcing the zoom-constraint is generally appropriate, there are exceptions to the rule. An example is generalizing regional labels at multiple scales. For this type of generalization the star approach [3] is a better fit. In this approach all scales are generalized from the same base dataset, which enables records to be selected only at intermediate zoom levels.

7 What we promised the reviewers

Changing geometry (tags: discussion) (copied from author-feedback) The question regarding changing the geometry type is very relevant and we have discussed it extensively while developing our framework. Let us first consider the semantics of changing geometry type. We can have a set-based semantics, in which a set of objects with given geometries are changed into a different set of objects (e.g., multiple objects are aggregated into one). Alternatively, we can consider a per-object semantics, in which individual objects remain identifiable throughout the generalization process, but their geometries change either by simplification (reducing the fidelity) or collapse (converting, e.g., a polygon to a point). Set-based semantics would imply changing dynamically the very formulation of the multi-scale filtering optimization problem. Given this complexity, we favor the per-object semantics, assuming that geometry changes are not significant enough to affect the solution. Under this assumption, a per-object semantics would be easy to incorporate in CVL through an extension to the generalize statement specifying a geometry modification function.

From an implementation point of view, such an extension can be supported in our algorithmic framework either as a post-processing step to the geometries before finalization, or as a geometry transformation step per zoom level. The latter alternative offers the more interesting opportunity of having multiple object representations on different zoom-levels, necessitating object versioning during generalization. Under the assumption above that geometry changes do not affect the optimization problem, this extension could actually provide us with potential performance gains, since it is faster to compute predicates over simple objects than complex ones. We plan to include in our paper a summarized discussion of this important issue of multiple object representations.

Running time (tags: discussion) (Copied from auther-feedback) It is an important observation, with which we agree, that the running time of the LP-based greedy algorithm is high. We implemented this algorithm as an alternative to the SQL-based SGA-algorithm mainly because it provides a bound on the solution quality. It should be noted that we have investigated other algorithms which could potentially have much better performance, albeit providing looser bounds on quality (ref [34] in the paper). These algorithms allow for different trade-offs between quality and performance, as well as careful reasoning about SQL-based implementations, to be explored. This is an interesting avenue for future work. We plan to add a discussion on this possibility to the paper, as well as address the other comments on your review.

8 Addressing selected comments in review

- R2 says: Section II: it seems you do not handle "regional labels" that might reappear at more general zoom levels. (Eg, "Europe" reappears when the camera pulls out far enough.) That should be made explicit in the text. (also asked by KMS people)
- R2 says: Not sure why the "AT X ZOOM LEVELS" clause is useful. Can you provide a motivating example? Current examples don't make the case. E.g., why "18" in Figure 4?
- R2 says: The mapping of conflict sets to set cover is not done convincingly. This mapping should be much more carefully written, with examples provided. Kostas:

We can put a bi-partite graph. Marcos: could put bi-partitite graph as in-paragraph figure, as it is space efficient.

9 TAIL

We have conflict sets because we want to satisfy constraints. We want to satisfy constraints because it is a common way to reason about feasible solutions to generalization... thinning and "The constraint method for solving spatial conflicts in cartographic generalization"

10 Question: Why map conflict resolution to an NP-hard problem?

Here we assume that using conflict sets is a good idea. The question whether that is true is a separate question, that we will also answer.

- Definition of conflict resolution: Given a set of conflict sets C, and a number $\lambda_c < |c|$ for each set $c \in C$, choose λ_c elements from each set c that will be "deleted"
- Global optimization version: choose λ_c elements of minimum total weight from each set c that will be "deleted"
- This optimization problem equals set multicover problem, plain and simple. I don't see that we are even "mapping" to this problem. It is the same problem?
- Is there another optimization problem (not minimizing total weight of elements deleted) that involves conflict sets and generates useful solutions to generalization?
- The SGA algorithm is an exact algorithm for another optimization problem (minimize for each set); but is that really the "right" problem to solve and why not?

10.1 Our answer

- Actually, for most instances the bulk of the time is spent finding conflicts, not solving the set multicover problem, because in the end we don't solve SMCP
- We apply approximation algorithms and heuristics which are exact solutions to various other problems that somehow relate to SMCP

11 Question: Why use conflict sets to model problem?

- Is there a "conflict free" formulation that would be just as "good" and not lead to solving an NP-hard problem?

- Conflict sets allow a generic way to express many natural user-defined constraints (proximity, visibility)
- Using conflict sets is a natural way to think about generalization
- Maybe there is another model that could express proximity and visibility constraints, and which is a natural way to think about generalization, and which leads to an optimization problem that has a polynomial time algorithm?

11.1 Our answer

1. We chose conflict sets because it is a natural way to think about generalization and because it allows users to formulate several natural spatial constraints (proximity, visibility etc)
2. We can not think of another model with these properties and which also leads to an optimization problem that has an efficient solution
3. Asking us to prove that such a model does not exist is a tall order...

References

[1] P. Alimonti and V. Kann. Some apx-completeness results for cubic graphs. *Theoretical Computer Science*, 237(1):123–134, 2000.

[2] K. Beard. Constraints on rule formation. *Map generalization: making rules for knowledge representation*, pages 121–135, 1991.

[3] T. Foerster, J. Stoter, and M.-J. Kraak. Challenges for automated generalisation at european mapping agencies: a qualitative and quantitative analysis. *Cartographic Journal, The*, 47(1):41–54, 2010.

[4] M. R. Garey and D. S. Johnson. The rectilinear steiner tree problem is np-complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977.

[5] L. Harrie. The constraint method for solving spatial conflicts in cartographic generalization. *Cartography and Geographic Information Science*, 26(1):55–69, Jan. 1999.

[6] L. Harrie and R. Weibel. Modelling the overall process of generalisation. *Generalisation of geographic information: cartographic modelling and applications*, pages 67–88, 2007.

[7] S. Schmid. *Automated constraint-based evaluation of cartographic generalization solutions.* PhD thesis, Geographisches Institut der Universit{ä}t Z{ü}rich, 2008.

[8] R. Weibel and G. Dutton. Generalising spatial data and dealing with multiple representations. *Geographical information systems*, 1:125–155, 1999.