Optimisation of generalisation re-calculation using partitioning

Tomas Straupis*

iTree tomasstraupis@gmail.com

* corresponding author

Keywords: spatial data, cartographic generalisation, multiple representation, multiple scale maps, partitioning, continuous generalisation, generalisation partitioning

Abstract:

Expectation and demand by map users is calling for much more frequent updates to maps. In some cases map needs to be updated as soon as new data is collected, which in the rapid rise of the usage of remote sensing technologies and participatory on-line mapping (like OpenStreetMap) means almost real-time updates.

When creating cartographically sound maps for different scales, cartographic generalisation is essential. But generalising cartographic data requires a lot of resources and time which makes it impossible to update multi scale maps more frequently even when original raw data is being updated.

Time (t) needed to perform each generalisation operator could be expressed as:

$t=t_o*n_o$

where:

n_o - number of objects being generalised

t_o - average time required to generalise one object

There are two possibilities to decrease the time required for generalisation. The first one is reducing the time taken to generalise one specific object (or group of objects). Second one is reducing the number of objects being generalised.

This paper focuses on the second option.

When generalising the initial dataset for the first time, there is not much we can do to reduce the number of objects being processed. But when the original dataset is being periodically updated and we have information on which specific objects have changed, we can re-process (re-generalise) only the objects which have changed, thus reducing the number of objects processed and therefore total time taken for generalisation.

Identifying objects to be reprocessed is simple, when the generalisation operator works on a single object, a good example of that would be building simplification - this operator usually is not influenced by other nearby objects. Therefore it is possible to re-simplify the building geometry when and only when its original geometry has changed.

It is not that simple when generalisation operators such as amalgamation, displacement, typification etc. are used, which take into account surrounding objects. In that case we might need to recalculate generalisation not only for objects which have actually changed, but also their neighbouring objects, neighbours of the neighbours etc.

Proposed method is to calculate clusters of initial objects with a cluster distance being the same as that of generalisation operator maximum distance as well as an additional buffer of the same distance. For example if we're going to

amalgamate buildings closer than 10 metres, then we have to calculate clusters of initial objects which are closer than 10 metres. After the change of any object in that cluster we would have to recalculate the whole cluster - delete all generalised objects in an area of that cluster and generalise objects in the area of the cluster again. Then we would have to re-calculate such cluster with changes again to be used during the next update of the dataset, because if any object in a cluster has changed - the cluster itself has also probably changed (see Figure 2 for the process).

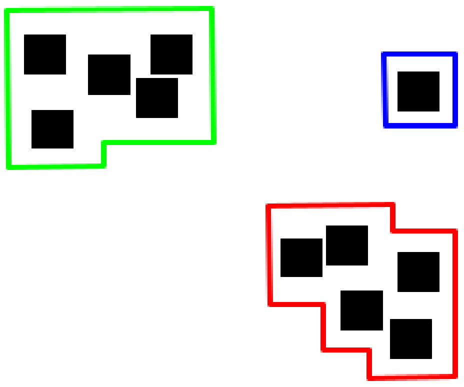
Figure 1. Clustering of original objects.

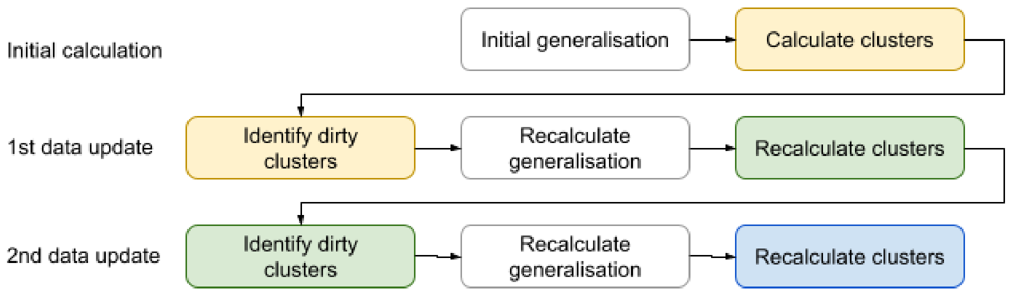

Figure 2. Cluster/Generalisation calculation/usage process.

As depicted in Figure 1, change (modification, deletion, adding) of any building in the green cluster will cause the following actions:

1. Mark the green cluster as “dirty” - set for re-generalisation.

2. Recalculate the buffered geometry of the green cluster given the change of its objects. Check if the green cluster from now on accumulates with any other existing clusters, if so - combine the clusters (combined cluster is also “dirty”).

3. Create clusters for any new isolated objects which have not so far been grouped into previously created clusters and mark these clusters as “dirty” as well

4. Delete all previously generalised objects in (covered/intersecting) “dirty” clusters.

5. Re-generalise all objects covered by “dirty” clusters.

Final expected results are: define a method of generalisation partitioning in order to re-generalise only the required set of initial objects and evaluate the impact of using this method on time required for data set re-generalisation.