Automatic Generalisation for production

Nicolas Regnauld
Nicolas.regnauld@1spatial.com
Product Manager
1Spatial
Tennyson House, Cambridge Business Park
Cambridge, CB4 0WZ
United Kingdom

Introduction

This paper is not a research paper, but presents 1Generalise, an automated generalisation software that has been built by 1Spatial, as part of a software suite to manage spatial data (1Spatial 2014). 1Generalise was built on the back of recent successes in deploying bespoke generalisation solutions in a few mapping agencies. The first part of this paper is a short reflection on the main challenges to develop a generic generalisation system fit for production. The two following sections present the approach implemented in 1Generalise two respond to two of these challenges: providing a system which is easy to customise, and providing a system that can process large national datasets and then update them by propagating change from the source database.

Making a generalisation system for production

Automatic generalisation has slowly moved in the last few years from being mostly a research topic to being part of production systems in National Mapping Agencies.

The first mapping agencies that were able to benefit from such automation were those who had heavily invested in the technology, buying cutting edge software and training specialists to customise and extend them to produce the output they wanted from their data (Maugeais et al 2011, Stoter et al 2013, Regnauld et al 2013).

From there, like in any other industry, once the first custom systems are in place, the challenge is to make the technology available to a wider range of users, by reducing the total cost of ownership of these automated systems. With 1Generalise, 1Spatial brings to its customers a system which has all the ingredients to achieve this:

Powerful generalisation capabilities:

1Spatial has more than 20 years of experience in developing generalisation systems. The generalisation tools available in 1Generalise benefit from that experience. They benefit from a rich framework which provides explicit topology, an object database, and an optimisation engine. Explicit topology is very efficient to ensure that topological relationships between features are preserved during generalisation. An object database has many advantages, like allowing direct references between features. An optimization engine is useful to resolve a situation where it is very hard to propose a sequence of operations that will resolve it, but the criteria to assess the quality of the result are well understood. The optimisation engine available in 1Generalise comes from the AGENT

project (Barrault et al 2001), it uses a multi agent system to maximise the satisfaction of a set of constraints.

Easy to customise:

When it comes to generalisation, there is no 1-fits-all solution. Users have different data specifications, different product specifications, and therefore the process to derive maps from source data is specific to each case. An automated generalisation system has to be customisable to allow this, but what level of customisation is the most appropriate? The maximum customisability would be reached by opening up the system to let the user extend it programmatically, by developing their own generalisation tools using a programming language and a set of API giving access to existing functionalities. This however requires the user to build up and maintain specialist skills in at least two domains: cartography and programming. Previous experiences doing this have proved this to be very expensive, slow to put in place and risky. This would go against the aim to reduce the cost of ownership of the solution. Instead, we have opted for opening up the level just above, where the user accesses the logic used to combine the tools to achieve the generalisation they want. 1Generalise controls the low level generalisation algorithms through a set of rule-based actions. An editor is included for the user to customise these actions. The rule syntax is easier to read and learn than a programming language (for someone who isn't a computer scientist). The aim is that the rule engine is used by the domain expert, the person who knows about the data, the end product (a map or a low resolution dataset) and the generalisation principles required to perform the transition.

Ready for production:

For a system to be used in a production environment, it needs to fulfil a set of non-functional requirements, relating amongst others to speed, resilience and scalability. These requirements are usually not the focus of research studies, although speed and scalability should be considered (and usually are). For a production system, these requirements need to be taken into account from the start, and be reflected in the architecture of the system. 1Generalise is built using the Java EE application platform, it can be deployed on a grid of processing nodes. This combined with a partitioning strategy implemented in 1Generalise provide scalability and resilience to the system. Adding more nodes to the grid will speed up the process (more partitions processed in parallel). Processing independent partitions increases the resilience, as if an incident occurs on a processing node (lost connection to the data base, data issue, software bug or hardware failure), only the partition involved needs to be reprocessed, without affecting others.

1Generalise uses a relational database to store both rules and data (Oracle and soon PostgreSQL), and runs inside an application server (Oracle WebLogic or Red Hat Wildfly).

1Generalise is configured and triggered either through a Web interface or using APIs: REST APIs to configure 1Generalise or SOAP API for running generalisation jobs. These API are used by the web clients that provide the UI to use 1Generalise, and can be used to integrate 1Generalise in a larger system.

Controlling the generalisation logic through rules

In this section, we present how 1Generalise can be customised by editing the rules that control the generalisation. In 1Generalise, the generalisation logic is stored in a **Flowline**. A flowline is a preconfigured generalisation process. To derive a specific type of map, it generalises a predefined set of themes where a theme is typically a type of geographic feature (roads, buildings rivers). Flowlines are organised into subflows which are a sequence of actions associated with a theme. Each action performs a logical generalisation operation to features within that theme. Actions and subflows are reusable components which are controlled using parameters. User data is processed by a flowline through setting up a classification and a profile. This is explained in (Regnauld 2015). Here we concentrate on writing new actions. Ultimately, a flowline is a long sequence of actions, over which the flowline designer has full control. The complexity of each action varies from triggering a simple independent algorithm (delete small features), to performing contextual operations (like displacement or amalgamation) or even triggering tasks involving our optimisation engine AGENT.

1Generalise comes preconfigured with ready to use flowlines, plus a flowline designer interface for authoring new flowlines or editing existing ones. The intention with these preconfigured flowlines is to provide a starting point to the user, where they can see some initial results, and gradually change bits in the flowline until they get to the results they want. This follows the main agile principle, which is to improve the solution step by step rather than building a large process upfront and testing it at the end. This ensures that the most important things are addressed and no time is wasted trying to do things which add little or no value to the product.

To demonstrate how 1Generalise can be customised by adding and editing actions, we take an example. We start from a flowline that we have written for the European Location Framework project (http://www.elfproject.eu/), we add an action to it. The data used come from the Dutch Kadaster, they were supplied for testing the development of the ELF generalisation services. The results are shown using the 1Generalise viewer, which lets the user see side by side intermediate results during the execution of the flowline. The user can select the point in the sequence at which the data is displayed in each screen, for example the source data on one side and the final result on the other, or just before and just after a particular action to analyse what it has done. The results is also shown using QGIS to style them with cartographic symbols. QGIS directly connects to the Oracle database used as source and target by 1Generalise.

Figure 1shows the result of the flowline in the 1Generalise dual viewer. The source data is shown on the left window and the generalised result is shown on the right. We can clearly see in the generalised window that the building on the right overlaps the landcover. We want to add a rule that prevents this from happening. The best option would be to write an action that moves the boundary of the landcover away from the building. However, for the purpose of demonstrating how to write an action to improve the result, we will use a simpler solution and add to the ELF flowline a step which clips landcover features to prevent them from overlapping building features.

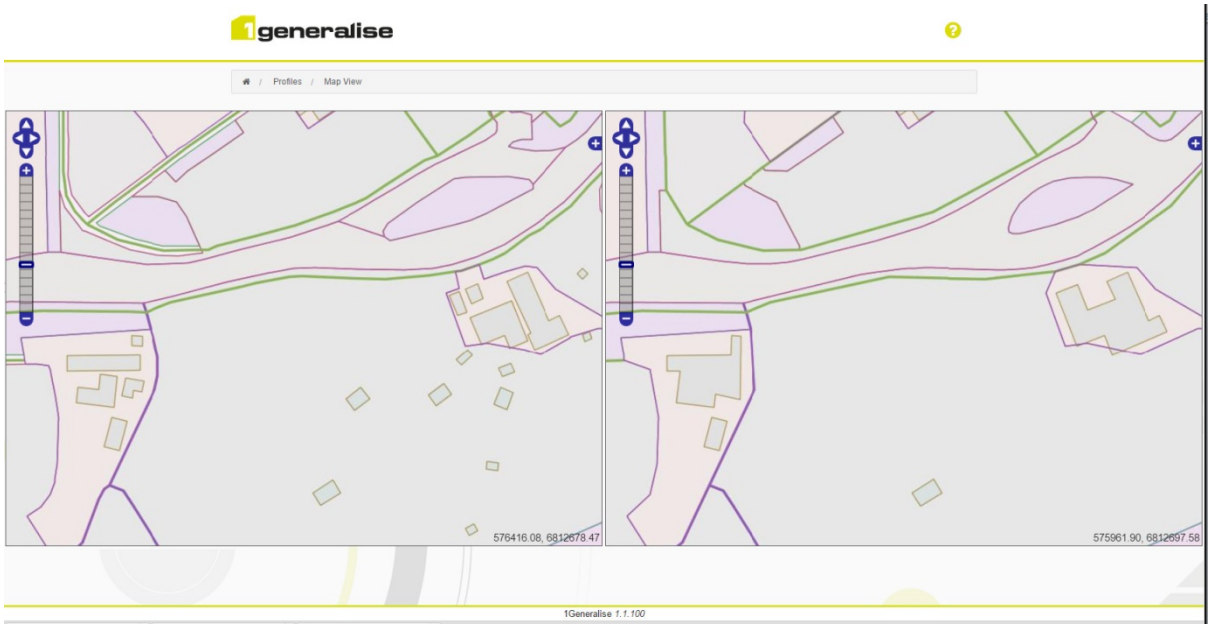

Figure 1: Flowline without clipping. Left screen: source data, right screen: result without clipping

Figure 2 shows the action we have added to perform the clipping. The action uses themes defined in the flowline (LANDCOVER_TGT and BUILDING_TGT). The first line is a loop through all the features of the LANDCOVER_TGT theme, they are temporarily named lc1. For each one, it loops through all the BUILDING_TGT features that overlap lc1. For each of these buildings (temporarily named bldg1), the geometry of the building bldg1 is subtracted from the geometry of lc1. Note that the line starting with "?" is a test, and that the 3 following lines are the three parameters used to define that test (geom1, spatial condition, geom2). Similarly, the line starting with "Let" is an assignment, followed by two lines to specify the two parameters recipient and value of that assignment. The recipient is the geometry of lc1, and the value calls a built-in function that performs the difference between two geometries. Built-in functions are where complex geometrical operations are performed. They are implemented in Java or C, and made available to the flowline designer, who can call them as and when needed. A rich set of built-in functions to perform generalisation operations such as simplification, smoothing, amalgamation, displacement, etc. are available.

Editing these actions is done through a UI that lets the user choose the elements of the action to be inserted, as shown in Figure 3 for inserting a loop.

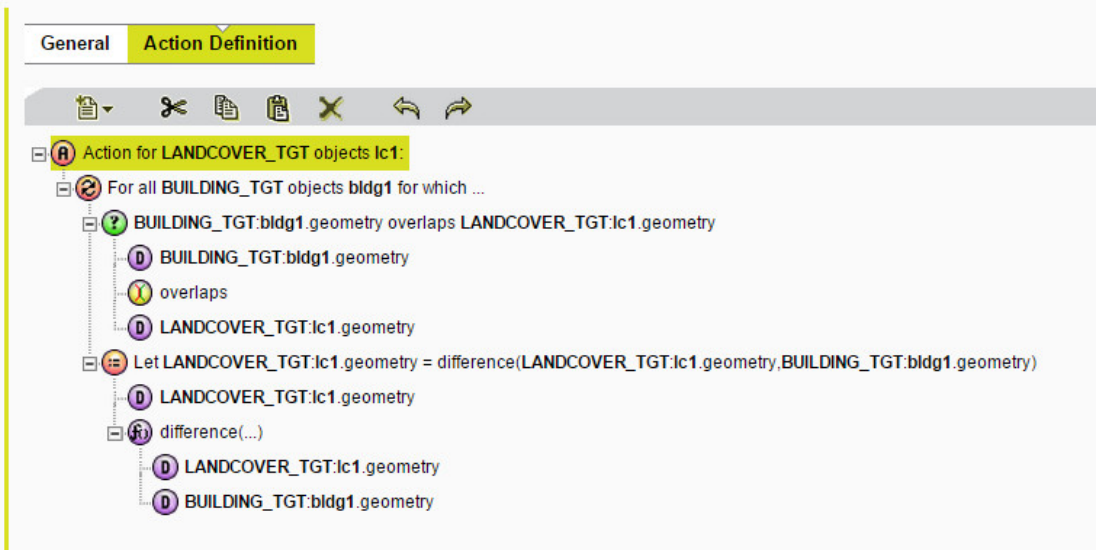

Figure 2: simple action to clip Landover features to avoid buildings

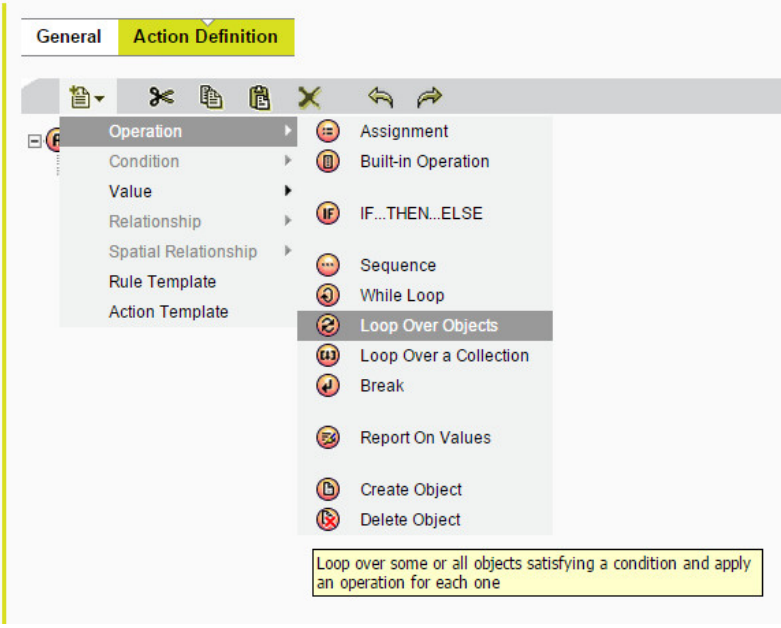

Figure 3: inserting a loop in the action through the UI

Figure 4 shows the data just before (left) and just after applying our new action. We can see that the landcover does not overlap the building anymore. Figure 5 shows the symbolised results in QGIS, the source data on the left and the results of the flowline with our action on the right.

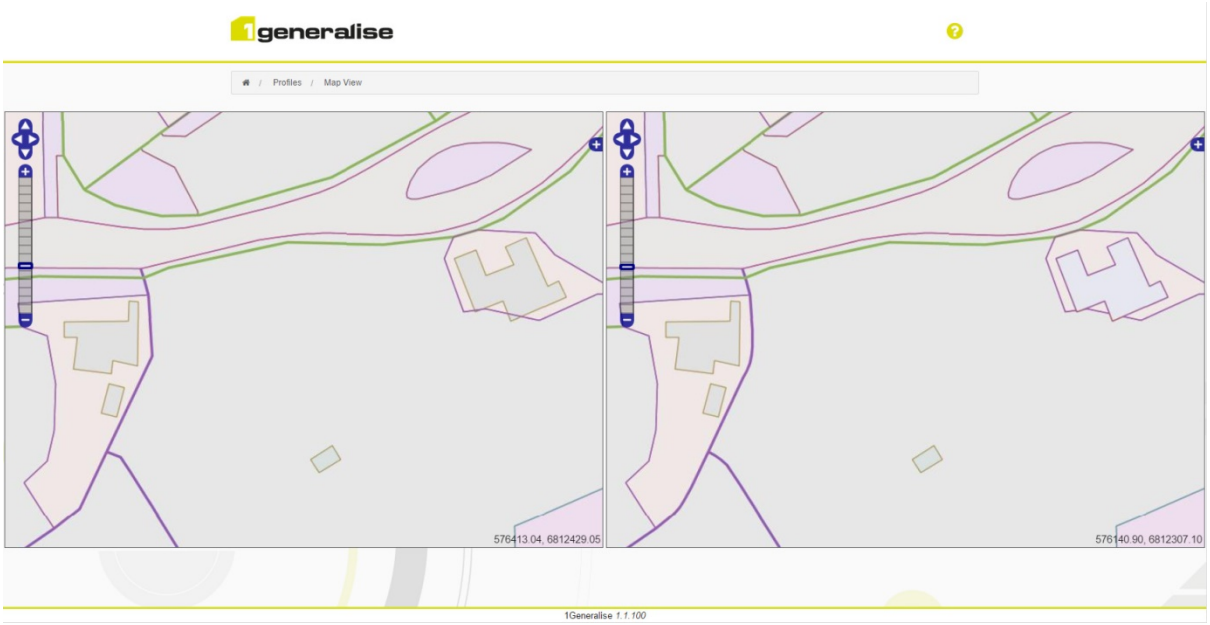

Figure 4: before (left screen) and after (right screen) clipping.

Figure 5: symbolised source data (left) and generalised data (right)

This has showed how writing a simple rule gives the user a fine control over the generalisation process, without needing to use a programming. The rule is edited through a Web browser and ready to use by 1Generalise as soon as it is saved.

Partitioning mechanism to support initial creation of national datasets and change only updates

While many actions are used to generalise features, others have different functions. Among those are actions that provide 1Generalise with a mechanism to partition the dataset, and use these partitions to control how the flowline loads the appropriate data and export the results in two different contexts: the initial creation of a generalised dataset with a national coverage and the propagation of changes to the target dataset.

Initial creation of large generalised datasets

In order to process large datasets efficiently, 1Generalise splits them into chunks that can be processed in parallel. The key to a successful splitting strategy is to minimise the number of occurrences of features crossing the boundaries. We use a number of strategies for that purpose:

- Use as partitioning features, those which are less often crossed by other features. The choice of the types of features to use to form partition boundaries can be configured in 1Generalise, but the default option is to use roads. Topographic data are often collected using rules that split features when they meet a road.
- Minimise the total length of the boundaries. To achieve this, we first aggregate the partitions created from our roads (or other features), until reaching an optimum data volume for our partitions. In order to limit the total length of boundaries, we also ensure that the aggregation mechanism generates large partitions which are as compact as possible.

Using geographic features is not always enough to partition a national dataset completely, as they may not reach the boundary of the dataset. This is the case near coastlines. Even assuming that a coastline feature exists, is continuous, and that no data beyond the coastline needs to be processed, using a combination of coastline and roads will not produce entirely suitable partitions. Roads don't meet the coastline, so a huge partition will get created along the coastline, potentially making an immense ring in the case of an Island like Great Britain. To avoid this problem, we have combined the geographical feature based partitions with a grid, in order to split the space around the edges of the dataset. The size of the grid is a parameter that can be tuned. As the grid may create very small partitions near the existing road partitions, a last step of merging these is applied. The main steps of this partitioning process are shown in Figure 6Figure 7.

Figure 6: Partitioning large datasets (1). Left: initial road partitions. Right: partitions aggregated

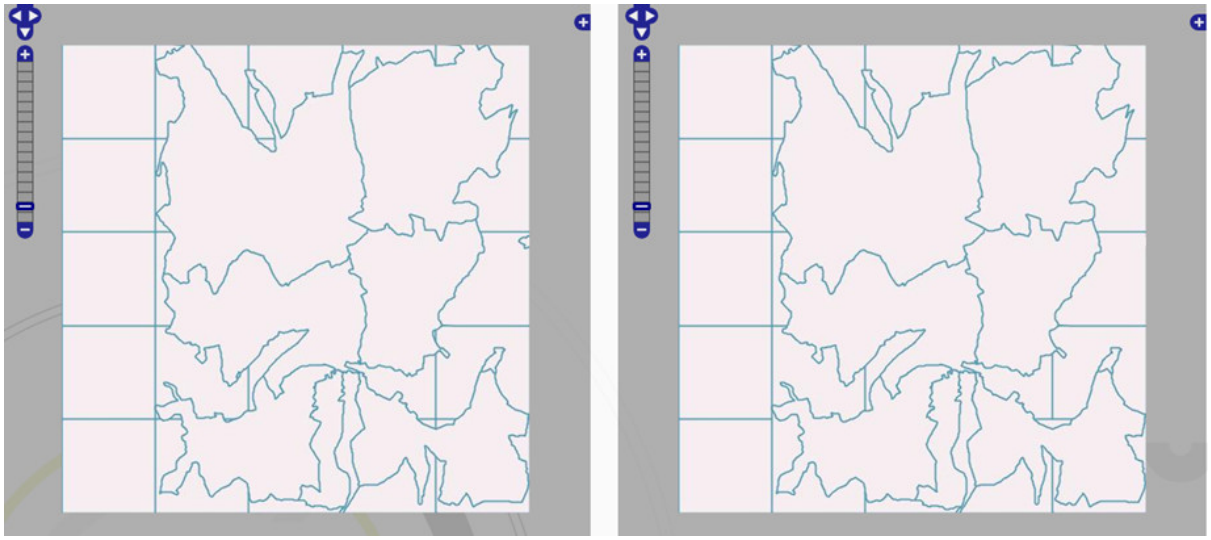

Figure 7: Partitioning large datasets (2). Left: grid applied to partition the space around the edge of the dataset. Right: very small partitions merged

Once the partitions have been created, the flowline carries on with generalising the roads that form the partition boundaries (only those). These will be considered fixed in the rest of the process. The last step of the flowline is to create a new job for each partition. This gets automatically picked up by the processing nodes available on the grid.

It is important to note that the ideal number of partitions is dependent on the number of processing nodes on the grid, and the average time required to process one partition. Too many partitions will increase potential problems at boundaries and will increase the total overhead incurred with creating new jobs. Too few will mean that some processing nodes might become available while no partitions are left to be processed and other processing jobs are still hours away from completing their current job on a large partition.

Change only update

Once a first version of a generalised dataset is available, 1Generalise offers the capability to update it by only reprocessing areas near the change which has occurred in the source data.

For a change only job, 1Generalise takes a geometry that encloses a cluster of change. An external component needs to identify these by analysing the database. Once 1Generalise receives the change, it performs the following actions:

1. Load from the existing target dataset partitioning features which are within a distance of the change extent.
2. Partition the dataset and identify the smallest set of target partitions that fully encloses the change extent. If none can be found, the process restarts at step 1 with a larger buffer.
3. Identify the matching source partitions in the source dataset.
4. Load data within and around (for context) these source partitions.
5. Generalise the data.
6. Clear the old data which are fully contained inside the target partitions.
7. Export only the generalised data that are fully contained inside the target partitions

Figure 8 shows the 1Generalise viewer at step 2 on the left window (target partition identified) and the result of step 4 on the window on the right (source data loaded in the partitions plus a buffer). The red square materialises the change extent that was passed to 1Generalise. Figure 9 shows the result obtained viewed with QGIS. In order to visualise the result, an action was added in 1Generalise to set an attribute on all features that it exported. The initial creation job set them all to 1, and the subsequent change only update job set them to 0. QGIS has been setup to show features with a value of 1 for this attribute in red and those with a value of 0 in green. As a result, it is easy to see which features have been updated by 1Generalise, all those contained in the smallest set of partitions enclosing the change, excluding features on its boundaries.

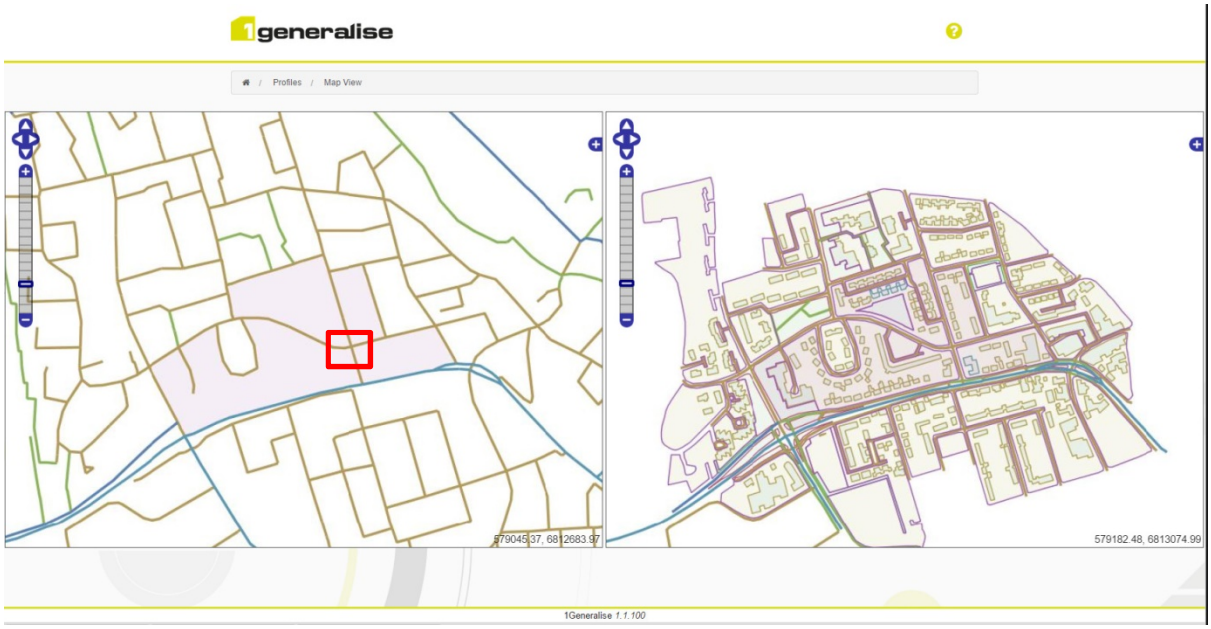

Figure 8: Change only update: Identifying target partition (left), loading source data (right)

Figure 9: reseults of change only update job (green features have been updated)

Conclusion

In this paper, we have shown some of the main features of 1Generalise: the easiness of customising the rules that perform generalisation operations, the ability to perform generalisation of large dataset and of propagating updates to a previously generalised dataset. This makes 1Generalise a powerful application to perform automated generalisation in a production environment. It is currently used in production at Ordnance Survey of Great Britain (Howland 2016).

1Generalise also has the unique characteristic of coming with a predefined flowline which is already setup to run in both national creation and change only update mode. This allows users to reuse that flowline and use it as a base to make their own, only changing the actions that relate to the actual generalisation rules. Their flowline will then be useable in both national creation and change only update modes, without them having to setup these complex processes, but still having access to it if they need to change something there.

1Generalise is also very open and can be configured and run through Web service calls. This offers the potential for using 1Generalise in other contexts, for example on demand mapping services. While we haven't worked on it so far, we see this as an exciting research area, already very active (Mackaness et al 2014), which will lead to new ways of generating maps that serve a specific purpose.

This leads to questions about the potential use of 1Generalise in the context of research studies. 1Spatial is very open to it and would like to take the opportunity of this workshop to discuss with the research community how this could be done. We would like to understand how researchers would like to use it, to work out the best way for us to make it available and to provide adequate support. The presentation at the workshop will mostly focus on this.

References

1Spatial (2016) http://1spatial.com/products-services/1spatial-management-suite

Barrault M, Regnauld N, Duchêne C, Haire K, Baeijs C, Demazeau Y, Hardy P, Mackaness W, Ruas A, Weibel R (2001) Integrating multi-agent, object-oriented, and algorithmic techniques for improved automated map generalization. In: Chinese Society of Geodesy Photogrammetry and Cartography (eds) Proceedings of the 20th international cartographic conference, Beijing, 2001

Howland D. (2016) OSGB Multi-Resolution Data Programme (MRDP), 2nd ICA / EuroSDR NMA Symposium, Amsterdam 3/4thDecember 2015, http://generalisation.icaci.org/images/files/workshop/symposium2015/OSGB_-_Presentation_Abstract.pdf

Mackaness W, Nick Gould N, Bechhofer S, Burghardt D, Duchene C, Stevens R, Touya G (2014) Thematic workshop on building an ontology of generalisation for on-demand mapping. http://generalisation.icaci.org/images/files/workshop/ThemWorkshop/ThematicOntologyOnDemand_Paris2015.pdf

Maugeais E, Lecordix F, Halbecq X, Braun A (2011) Dérivation cartographique multi échelles de la BDTopo de l'IGN France : mise en œuvre du processus de production de la Nouvelle Carte de Base. In: Proceedings of the 25th International Cartographic Conference, Paris, July 3-8.

Regnauld N, Lessware S, Wesson C, Martin P (2013) Deriving products from a Multi Resolution database using automated generalisation at ordnance survey. In: Proceedings of the 26th International Cartographic Conference, Dresden, 25–30 Aug 2013.

Regnauld N (2014) 1Generalise: 1Spatial's new automatic generalisation platform. 17th ICA Workshop on Generalisation and Multiple Representation, Vienna, 23 Sep 2014.

Stoter JE, Post M, van Altena V, Nijhuis R, Bruns B (2013) Fully automated generalisation of a 1:50k map from 1:10k data. Cartography and Geographic Information Science, vol 14, Issue 1.