Geoinformatica (2009) 13:425–452
DOI 10.1007/s10707-008-0054-3

Automated processing for map generalization using web services

Moritz Neun · Dirk Burghardt · Robert Weibel

Received: 12 July 2007 / Revised: 4 March 2008 /
Accepted: 6 June 2008 / Published online: 27 June 2008

Abstract In map generalization various operators are applied to the features of a map in order to maintain and improve the legibility of the map after the scale has been changed. These operators must be applied in the proper sequence and the quality of the results must be continuously evaluated. Cartographic constraints can be used to define the conditions that have to be met in order to make a map legible and compliant to the user needs. The combinatorial optimization approaches shown in this paper use cartographic constraints to control and restrict the selection and application of a variety of different independent generalization operators into an optimal sequence. Different optimization techniques including hill climbing, simulated annealing and genetic deep search are presented and evaluated experimentally by the example of the generalization of buildings in blocks. All algorithms used in this paper have been implemented in a web services framework. This allows the use of distributed and parallel processing in order to speed up the search for optimized generalization operator sequences.

Keywords Map generalization · Data enrichment · Cartographic constraints · Combinatorial optimization · Parallel processing · Web services · Service oriented architecture

1 Introduction

Map generalization seeks to maintain and improve the legibility of a map after the scale has been changed. During the map generalization process various generalization operators are applied to the features of a map such as buildings, roads, rivers, or land cover patches. Examples of such generalization operators include algorithms for the simplification, smoothing, aggregation, amalgamation, merging, collapse, refinement, exaggeration, enhancement and displacement of cartographic features [22]. The ordering of a workflow has an influence on the results [5]. Different paths (operator sequences) can lead to different

M. Neun (✉) · D. Burghardt · R. Weibel
Department of Geography, University of Zurich, Winterthurerstrasse 190, CH-8057 Zurich, Switzerland
e-mail: moritz.neun@geo.uzh.ch

results not only within generalization [17]. Therefore, the operators must be applied in the optimal sequence, with the correct parameterization and their results must be evaluated in order to achieve an improvement of the map legibility in the generalization process. Thus, if the aim is to fully automate the map generalization workflow and minimize human intervention, there is a need for automated control and sequencing of generalization operators.

Several conditions have to be met in order to make a map optimally legible. These conditions can be formalized with so-called cartographic constraints [42]. Figure 1 shows an example of conflicts that arise during map generalization. The left picture shows the conflicts that are created when the width of road symbols is enlarged. The buildings then overlap with the road symbols. Furthermore, some of the buildings are too small and would no longer be visible at the target scale (right picture). Therefore different generalization operators must be applied to resolve these conflicts, and they must be applied in the proper sequence. The operators that lead to the result in Fig. 1 are building simplification, typification and displacement, in this order.

The methods presented in this paper use cartographic constraints for the selection and sequencing of different independent stand-alone generalization operators (i.e. generalization operators that work independently and are not influencing each other). Thus, manifold operators with different grades of sophistication (from simple algorithms to sophisticated, context-aware algorithms [30]) can be combined to jointly form a powerful workflow by uniting their strengths. We use a modular service-based architecture that allows the integration of different stand-alone generalization operators even from different generalization systems on different platforms [8], [25]. With this versatile service-based architecture it is possible to build modular generalization processes that can be flexibly extended by introducing additional generalization operators. Thus, having a relatively large set of different operators available as services the goal is to select the optimal operator sequence in order to reduce the constraint violations for a given problem. This calls for the use of combinatorial optimization techniques [19] that control and restrict the application of the various operators.

The focus of this work is on introducing and revising constraint-based combinatorial optimization techniques for the control of the generalization process. The goal is not to optimize a specific generalization task using one specific algorithm but to find and optimize a sequence of multiple generalization operators that are applied to an entire set of map features. In our case we apply stand-alone generalization operators onto complete building partitions derived from a trans-hydro-graph [37]. Thereby it must be taken into account that the applied operators can have a different granularity in terms of their behavior. Some operators induce only small changes, or changes that can always be undone, while others make quite radical and irreversible changes (see Section 3.4). We show that it is possible to

Fig. 1 Conflict due to road symbolization, solved through simplification, elimination and displacement (examples show the source scale, the generalized result at the source scale and at the final target scale)

create an automated generalization workflow that applies stand-alone generalization operators in an optimized sequence. The service-based architecture allows the coupling of the operator services together with supporting facilities, like the evaluation functions, for processing with arbitrary optimization strategies. This novel modular processing approach also makes it possible to use parallelization techniques in order to speed up computationally heavy processing.

After introducing cartographic constraints and reviewing current automated generalization approaches (Section 2) the paper describes the constraints and the cost function used in the optimization techniques (Section 3). In Section 4, the different optimization techniques used in this work, including hill climbing, simulated annealing and genetic deep search are explained. In Section 5 the implementation of the workflow control in the service-based architecture is described and the improvement of the processing performance using parallelization is demonstrated. Finally, results of tests of the different optimization techniques are presented (Section 6) and discussed (Section 7). The paper ends with conclusions and an outlook (Section 8).

2 Background

2.1 Cartographic constraints

Conventional and automated map generalization both share the same basic objective, which is to ensure the legibility of a map for the map reader. Conditions that have to be met in order to make the map legible can be formalized with so-called constraints. Examples of cartographic constraints are those listed in Section 3. The concept of cartographic constraints was originally adapted from computer science to map generalization by [6]. Constraints received special importance in cartography through the application of intelligent agents in the area of automated generalization 0. Following the results from AGENT project [4], [32] constraints define a final product specification on a certain property of an object that should be respected by an appropriate generalization. While measures only characterize objects or situations [30], without considering cartographic objectives, constraints evaluate situations with respect to the formalized cartographic objectives. Thus, the constraints check whether the objects or situations are also in a cartographically satisfactory state.

Constraints can be used to describe object characteristics and relationships according to the requirements for a specific map scale and type. [3] proposed three types of assessment functions to determine the quality of cartographic generalization. The first type includes characterization functions, which characterize the geometrical and structural properties of single features or groups of features by means of constraints. The second type are the evaluation functions, which compare the states of the features before and after generalization. The third type includes the aggregation functions, which are used to summarize the individual evaluation results. Similar to this methodology, our approach aims to calculate a global cost function as proposed for the constraint space by [9].

2.2 Automating the generalization process

Automated control of the generalization process can be achieved using different approaches. [16] review existing methods, including simple batch processing, condition-action modeling, and finally sophisticated constraint-based techniques.

Static generalization workflows can be executed as batch processes. Here, no conditions can be applied and the parameters as well as the sequence of the operators are predefined. The modeling of conditional workflows within a generalization system is shown for example by [26]. For such rule-based processing approaches a human expert must formalize the cartographic knowledge into conditions and thus explicitly define the relation between conditions and actions and their order of processing. The selection of such rules is addressed for example by [29]. This selection of rules is also part of the knowledge acquisition process [13]. [14] propose machine learning techniques for deriving the cartographic rules. The combination of constraint-based evaluation and machine learning techniques for the knowledge acquisition is shown by [24].

Using constraint-based techniques the cartographic knowledge is captured in terms of conditions that have to be met. To satisfy these constraints optimization techniques may be used. The goal is always to minimize the violation of the constraints. The constrained based methods can be further subdivided into complex techniques which perform different operations simultaneously, as opposed to methods that use constraints to chain specific algorithms that perform one operation at a time [23]. Examples of the first category are for instance least squares adjustment [15], [33], energy minimization [7], [2] or simulated annealing [40], whereas the AGENT approach [27], [31], [32] belongs to the second one. The AGENT approach tries to minimize the constraint violations for map features represented by autonomous software agents. The agents are trying to find an optimal state with minimal constraint violations. Combinatorial optimization methods [19] also try to find an optimal state. [39] are using iterative improvement for the displacement of buildings. In [40] an iterative improvement approach with building displacement, scaling and elimination is shown. This approach uses simulated annealing in order to find a global constraint violation minimum.

3 Constraints for evaluating map partitions

In this paper the development and implementation of a processing service for the generalization of individual buildings is presented. Therefore the legibility conditions focus exclusively on geometrical and structural aspects. For the characterization and evaluation of the generalization state of a building eight constraints were defined (see also Fig. 2). The constraints can be subdivided into two major categories. On the one hand there exist constraints that describe concrete conditions that have to be met. These constraints actively

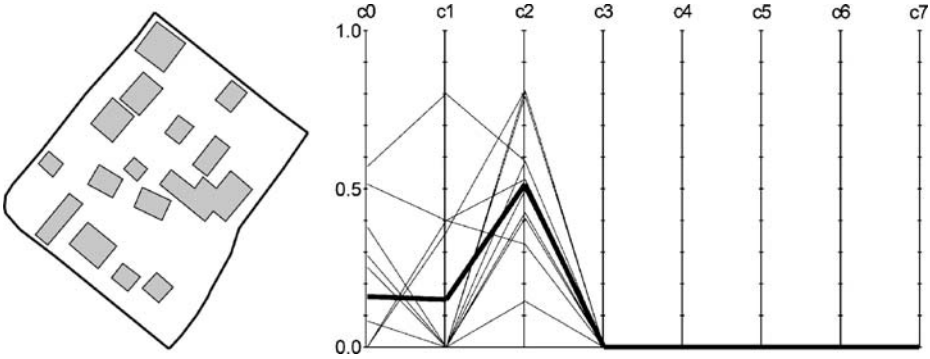

Fig. 2 Constraint violation before generalization

direct the required changes in order to be fulfilled and are therefore termed 'active constraints' [36]. The following four *active constraints* were used:

- the building must have a minimum size in order to be visible (acronym: MinSize; number: c0)
- the edges of the buildings must have a certain length in order to be visible (EdgeLength; c1)
- the buildings must be separated by a certain distance in order to be distinguishable (MinDist; c2)
- the buildings must have no parts which are not visible (LocalWidth; c3)

On the other hand there exist constraints that have the function to limit or prevent change. As they are only passively triggered, they are termed 'passive constraints' [36]. These constraints are responsible for the preservation of feature characteristics and relations. In the initial state of the map they are satisfied but during the map generalization process the may be violated. For example, the position constraint (DiffPos) is violated when a building is displaced (see Section 3.4). The following four *passive constraints* were used:

- the position of the building should be preserved as far as possible (DiffPos; c4)
- the number of edges of the building should be preserved as far as possible (DiffEdgeCount; c5)
- the width/length ratio of the building should be preserved as far as possible (DiffWidthLen; c6)
- the orientation of the building should be preserved as far as possible (DiffOrientation; c7)

Additional constraints could be inserted if needed in order to better define the desired map properties. The selected constraints can be established for every building. The minimum distance constraint (MinDist), however, analyzes the neighborhood of the building consisting of other buildings or nearby roads. All other constraints used solely focus on the geometry of an individual building. Depending on the concrete scenario also other constraint types could be used such as a semantic constraint in order to ensure a correct treatment of important features.

3.1 Map partitions

Partitions subdivide the map space in such a way that generalization tasks can process and evaluate them independently. This is an important preliminary step for the generalization of seamless data sets instead of map sheets. The partitions try to subdivide the data into coherent parts, hence isolating the generalization tasks. Map features inside a partition are assumed to be independent of changes made in another partition. Features forming the border of a partition should be static and should not change much during generalization. In our experiments we used the trans-hydro-graph as proposed by [37] for deriving partitions. Within other scenarios (e.g. involving other feature classes, outside the urban environment) other partitioning strategies may be more appropriate.

The trans-hydro-graph describes a structure derived from the transportation and hydrology networks. It can be used to isolate the task of building generalization since buildings must stay inside of its faces. The faces (or partitions) are formed by urban blocks and are usually small, containing between one and 80 buildings in our examples. Thus, they allow faster and parallelized execution of context dependent generalization algorithms as the data structures used do not become too large. The result of this explicitly established partonomic relation is a list of groups, whereby each group may contain any number of

buildings. Thus, a building partition (urban block) can also be seen as a meso-object [31]. Every partition can be characterized by a number of constraints which define an ideal cartographic situation. The constraints describe the fulfillment of a condition for every feature (i.e. building in our case) in the partition. Additional group constraints describe conditions for an entire partition. For instance, the minimum distance constraint (MinDist) controls the required distance between buildings inside a partition and between the buildings and their surrounding roads. The evaluation of a generalization result is always carried out for an entire partition in our current approach, against the constraints of all its buildings as well as its group constraints. Therefore, a cost function with weights for the different constraints is used (see Section 3.3).

3.2 Constraint visualization with parallel coordinate plots

For representing and analyzing the state of a cartographic situation the violations of constraints can be represented by so-called generalization state diagrams [4], [31]. For the visualization of large numbers of features with their associated constraints we propose to use parallel coordinate plots (Fig. 2). The parallel coordinate plots represent n cartographic constraints with their degree of satisfaction. The axes are scaled to the interval [0, 1], whereby a value greater zero means that the constraint is violated. The constraint values are equivalent to the so-called severity used in the generalization state diagrams.

Figure 2 shows the constraint violations for an urban block before generalization. The thin lines in the plot show the constraint violations for every individual building, the heavy line shows the average constraint violation for the entire urban block. Note that the constraints are not weighted.

Figure 3 depicts the constraint violations after generalization. This example shows that the generalization process has generally reduced the violation of the various constraints. In comparison with Fig. 2 it is also noticeable that in order to reduce the violation of constraints c0 to c3 the preserving constraints c4 to c7 had to be violated. The importance of the individual constraints and thus a possible weighting is expressed by means of a cost function.

3.3 Cost function

Automated generalization can be seen as an iterative process between conflict analysis and conflict resolution [30], [41]. Thus, the goal of generalization is to minimize the existing

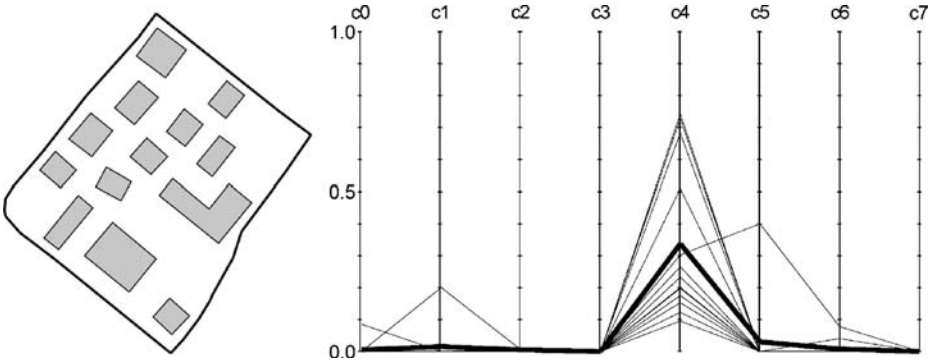

Fig. 3 Constraint violation after generalization

constraint violations without creating new ones. The difficulty stems from the fact that cartographic situations are connected to a set of constraints that partially work against each other. Examples are the constraint of ensuring minimal distances between objects (MinDist; c2) that works against the constraint that seeks to maintain positional accuracy (DiffPos; c4); or the need of reducing details (LocalWidth; c3) versus the constraint of preserving the original shape of buildings (DiffEdgeCount; c5). The goal of automated generalization, then, is to find a good compromise between all these several constraints. Thus, a minimal equilibrium between all n constraints must be found. This can be expressed by a cost function that is, as for our experiments, just the simple average of all constraint values. In order to favor and punish certain constraints a weighting can be applied to them when calculating the cost function:

$$\mathbf{Cost} = \sum (\mathbf{Constraint} * \mathbf{weight})/n$$

The weights are scaled to [0, 1] and add up to 1. The constraints and their weights are the only parameters that can be adjusted in order to modify the current system. For example the positional accuracy (DiffPos) could be weighted lesser or more depending on the map type. Especially the weighting of the passive constraints (the preserving constraints c4 to c7) can be used to tune and modify the selection of generalization operators. Weighting the passive constraints too low would result in a complete deletion of all buildings that are creating a conflict. In contrast high weights for the passive constraints would prohibit any changes, as every change would only increase the cost further. In our experiments we have found that equal weights for all four passive constraints work well with our optimization approaches.

A careful balance between the two constraint types is very important. The presented additive cost function does not solve the problem that different constraint values might have very different behaviors in terms of linearity. A more advanced cost function would take into account that one constraint might be transformed logarithmically while others are simply kept linear. Therefore a better understanding of constraints and of their behavior will be needed in the future [12]. However, optimizing the cost function and the constraint weighting is not the focus of this paper.

Constraints are used to validate the result of generalization operators and thus to select the appropriate operator sequence. Obviously, this validation can only validate what is formalized in the constraints. The results of the optimization techniques presented here can always only achieve the quality that is formalized by the constraints. Thus, the future correct and complete formalization of constraints 0 will be crucial for the correct validation of the result.

3.4 Effects of generalization operators

Typical generalization operators for our example of building generalization include simplification, exaggeration, aggregation, elimination, and displacement. These algorithms focus either on the removal or on the geometrical transformation of buildings. In automated generalization a particular generalization operator, e.g. building displacement, can be realized with different algorithms based on different solution approaches. Some operators have a narrowly defined functionality or are only applicable to specific feature classes, while others combine different functionalities. For instance, building typification, as realized in the algorithm proposed by [11] combines elimination, simplification,

exaggeration, and displacement as it replaces a group of buildings by a placeholder. In this context the typification operator is comparable to an aggregation algorithm.

It is important to note that there are two types of generalization operators which are separated by a fundamental difference in terms of their granularity. The first group of operators, including elimination, aggregation, typification and simplification, is applied in rather coarse grained and discrete steps, usually removing entire features or details of features. These operators often generate results that are absolute and irreversible. In contrast, the fine grained, more continuous operators are only making slight changes. It is important to note that these changes usually are reversible, as they are usually applied in iterative algorithms. A prominent example of such a fine grained reversible operator is feature displacement where positional changes can be redone in a later step. Likewise, the enlargement (exaggeration) and shrinking of features can be redone in a later processing step. This fundamentally different behavior of generalization operators is important when sequences of operators are created. An initial elimination of a map object can never be redone even if after several other generalization steps there would be sufficient space to retain the object. The displacement of features, however, can be reverted if the inverse displacement is applied in a later step.

Figure 4 shows the generalization operators that were used for the constraint optimization experiments. For every operator the result and the constraint violations are shown and can be compared with the 'initial situation'. Note that the sequence shown in this figure serves merely as an example to illustrate the various generalization operators. The values and behavior of these constraint violations may be very different for other building partitions. The implementation of the generalization operators shown here is described in more detail in Section 5.2.

It can be seen that the different operators are having quite different influences on the evolution of the constraints. In the initial situation the minimum building size (c0), the

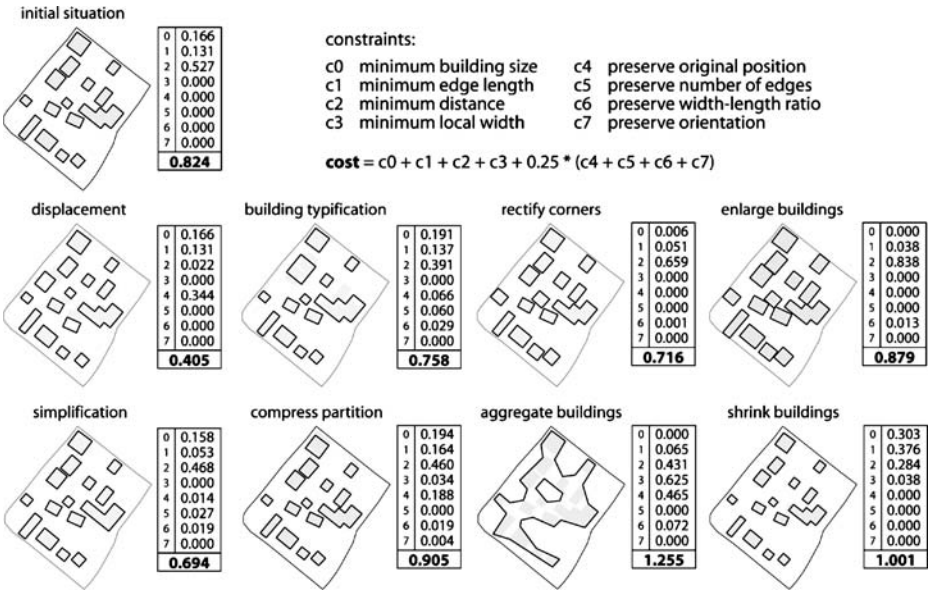

Fig. 4 Constraint violations for an initial map situation and after the execution of several generalization operators

minimum edge length (c1) and the minimum distance (c2) are violated. Operators for fulfilling the minimum size constraint are ‘enlarge buildings’, sometimes ‘enlarge to rectangle’ and under special conditions ‘aggregate buildings’ and also ‘building typification’ as they usually are removing the small buildings. For the minimum edge length constraint (c1) as well as for the minimum local width (c3) the ‘simplification’ is the most obvious operator but also the before mentioned enlargement operators tend to improve the situation. ‘Displacement’ reduces the violation of the minimum distance constraint (c2) but the original position cannot be preserved (c4). If not enough space is available to displace all buildings sufficiently also other operators like ‘compress partition’ and ‘shrink buildings’ or operators that reduce the number of buildings such as ‘building typification’ and ‘aggregate buildings’ may help. These operators may, however, influence the preservation of the original position (c4), the number of edges (c5), the width–length ratio (c6) and the orientation (c7).

4 Processing strategy—methods

This paper proposes and evaluates combinatorial optimization strategies for the automated selection and chaining of generalization operators, with the aim of applying them in the optimal sequence. The strategies are based on the capability of formalizing goals and requirements as constraints in order to evaluate the results of different generalization operators. Operator sequences are created by iteratively executing all the available generalization operators and then selecting one or more results using a heuristic in order to continue the process, again executing all the operators. The search algorithms used in this paper are *hill climbing*, *simulated annealing* and *genetic search* (cf. Section 4.2)

The generalization operators used in the processing strategies are completely independent from each other. Thus, they are behaving like a black box that receives input data with parameters and simply returns a result. Note that this stand-alone behavior is an intrinsic characteristic of using a service-based architecture. Through the continuous evaluation of these results the processing system is basically self-adapting. The operators are exchangeable and new operators can be added without having to make any changes; again, a benefit of using web services.

4.1 Search space

With a set of operators there exists a large space of possible operator sequences which can be pursued iteratively. The aim is to find an optimal result in this global space of possible solutions. Thus, global optimization addresses the task of finding the best operator sequences. These combinatorial problems are NP-hard [14]. In the experiments presented below (see Section 6) we use eight operators and a maximum sequence length of 20 steps, resulting in a search space that has at most 1.31E+18 (derived from $\Sigma\#\text{operators}^{\text{depth}}$) possible solutions. Thus, for the illustrations in this section a smaller search space is used. Figure 5 shows such a search space of possible results for the processing with three operators and a maximum sequence length of three. At least one of the 40 possible states (initial state, 12 intermediate and 27 final states) has a minimal cost.

An example of the search space for a sample urban block can be seen in Fig. 6. Each line-up shows the costs of one of the 27 possible operator sequences with length three. At the root of the search tree (depth 0) the costs are the same for all sequences. It can be seen that quite different sequences can lead to minimal costs (e.g. sequences 16 and 25) and even

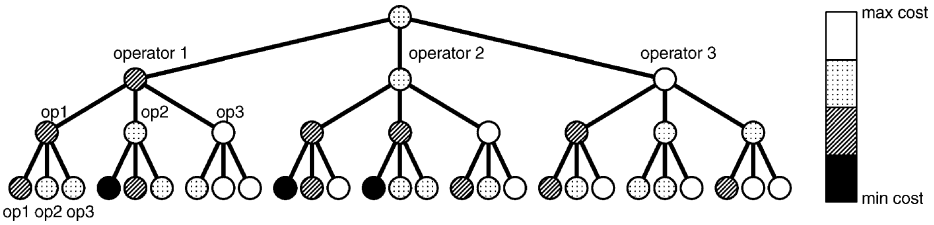

Fig. 5 Search space for combinatorial optimization with three operators

some others result in a near optimal result (e.g. sequences 4, 8, 10, 11, 13 and 22). But it can also be seen that the wrong sequence or operator selection can lead to quite a bad result (e.g. sequence 18) and that the optimal result in one sequence is not always achieved in the final step (e.g. in sequence 6 and 18). The two lowest minima for this example (sequences 16 and 25) are both in quite irregular groups surrounded by rather bad results with high costs. The sequences 1 through 9 have all rather minimal costs as they all start with a displacement. Here the sequences 4 and 8 lead to two local minima designating the minimal costs for sequences starting with a displacement.

Figure 7 shows the outcomes of three examples taken from the sequences of Fig. 6. Sequence 4 shows a solution that is acceptable, but there are still many small buildings that violate the minimum size constraint and some violations of the minimum distance constraint remain. Sequence 16 shows the best result which is achievable with only three operators. The number of buildings has been reduced slightly but there are still some minor violations of the minimum distance constraint. Sequence 18 proceeds in the first two steps similarly to sequence 16 but the final typification removed too many buildings, thus inducing excessive changes in the resulting map. In particular the constraints 'original position' and 'number of edges' (due to the removal of complete buildings) are violated severely. Thus, the final cost is much higher than the cost of the two other sample sequences.

It is important to note that the examples shown in Figs. 6 and 7 might not be cartographically perfect solutions. This is due to the fact that only three generalization

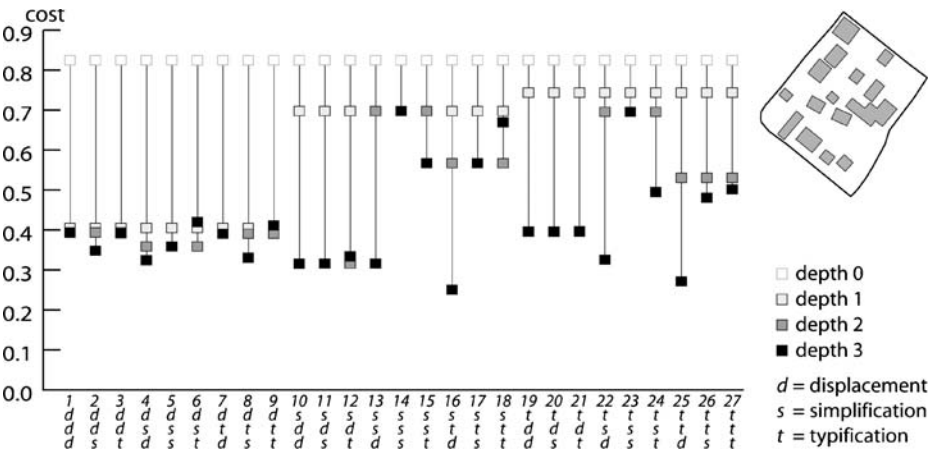

Fig. 6 Possible cost outcomes from a search space with three operators

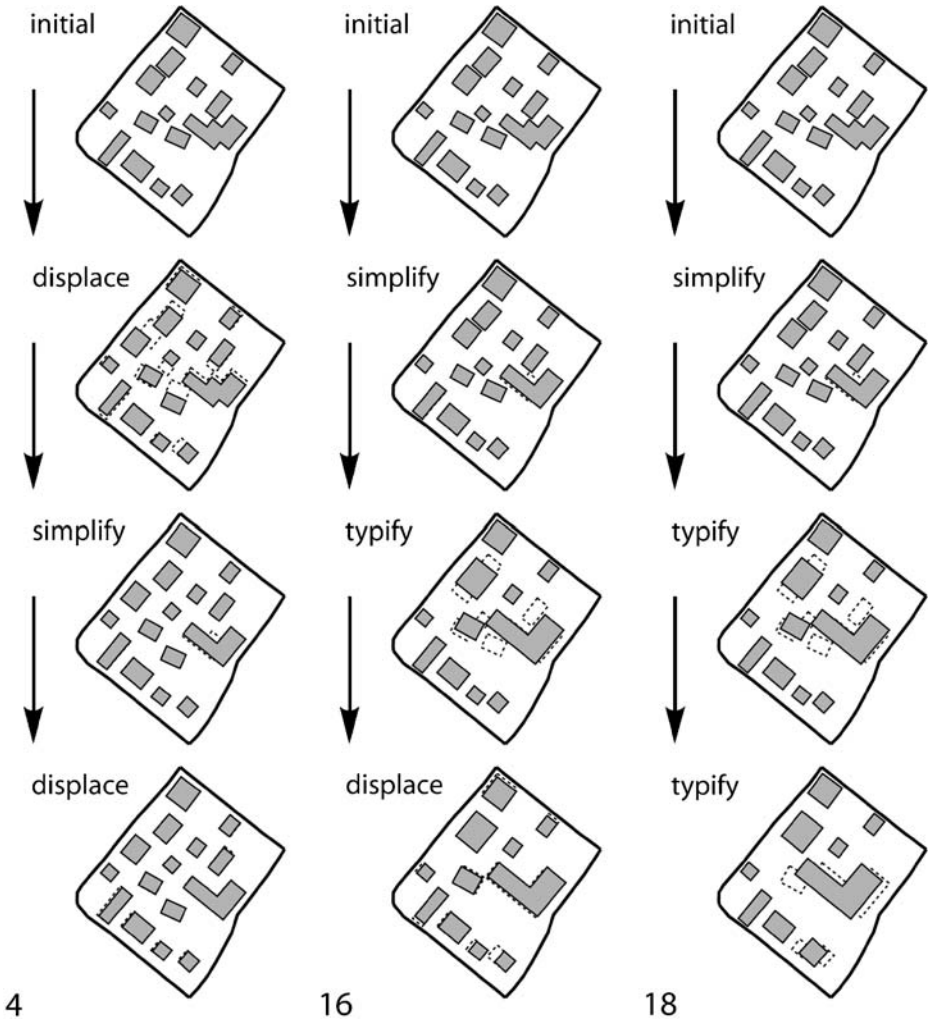

Fig. 7 Result examples for operator sequences from Fig. 6

operators with limited capabilities were used. The examples are simply used to illustrate the large diversity of results that are obtained when generalization operators are applied in permutated sequences.

In the search space example shown here three operators and a maximum sequence length of three was assumed. Thus, if all operators are used, every single one can only be used once. As shown in Section 3.4 there exist different types of operators. Some operators, including simplification, rectification or enlargement have to be executed only once. On the other hand, the typification operator may be used multiple times if the amount of buildings needs to be reduced substantially. Displacement in particular might be used at multiple points throughout a generalization process: In the beginning to remove distance problems, but also after a typification, aggregation, or enlargement of buildings in order to solve distance problems created by the preceding operators. Thus, the maximum sequence length is normally longer than the number of available operators. In the experiments reported in

Section 6 a sequence length of 20 was used with eight operators. It turned out that the average sequence length is between 6 and 11 depending on the search algorithm used.

4.2 Search algorithms

As shown above, even a moderately large number of operators and moderately long maximum sequence length result in a huge global space of possible operator sequences. A brute force approach would test all the sequences, forming a tree of results in order to obtain the best result (see Fig. 5). This *deep exhaustive search* is definitely too slow as it would result in millions of executions of every generalization operator for only a single building partition. Thus, a heuristic is needed as a simplifying search strategy in order to find a sequence with sufficiently minimal costs.

4.2.1 Hill climbing

The most obvious search strategy is the hill climbing approach. During the iterative optimization cycles always the currently best result is taken to proceed. Hill climbing approaches are proposed, for example, by [4]. This search algorithm assumes that at the beginning certain constraints are violated. The goal in the iterative processing cycle is to reduce these constraint violations as much as possible without violating others too much. Thus, only cost improvements are possible. A sequence which starts off with first increasing the cost in order to enhance the outcome of a following operator is not possible.

The basic processing cycle of the hill climbing algorithm is as follows:

(1) compute the constraint violations (value *cost before*) for the *current data*
(2) execute every available operator with a copy of the *current data*
(3) compute the constraint violations (value *cost after*) for every operator result
(4) if the *cost after* of the *best result* is lower than *cost before*: keep the *best result* as *current data* and the *cost after* as *cost before* and then continue with step 2 otherwise: return the *current data* as the final result of the processing cycle

This hill climbing approach is straightforward to implement and performs quite fast as only one sequence has to be computed per building partition. Figure 8 shows that the hill climbing approach always takes the best current result in order to proceed. Therefore it can get caught in a local minimum instead of finding the global minimum.

In Fig. 6, sequence 4 corresponds to the hill climbing approach. The result in this example is not perfect but still acceptable; this corresponds also with the experimental results over a larger number of building partitions reported in Section 6.

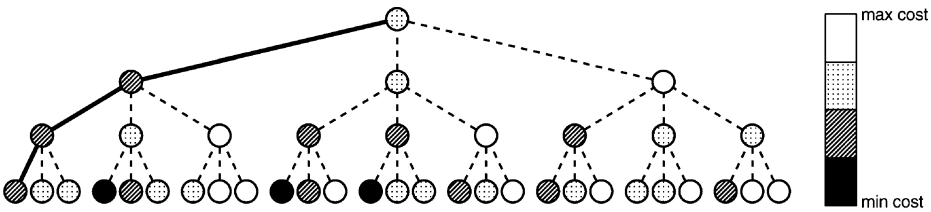

Fig. 8 Hill climbing algorithm, proceeds at every step with the best result (triplets ordered by cost from *right* to *left*)

4.2.2 Simulated annealing

Simulated annealing (SA) approaches try to reduce the likelihood of getting caught in a local minimum. Instead of always taking the currently best result also an inferior result or even a result which increases the cost can be taken with a probability *P*. The probability *P* is continuously decreasing over time so that at the beginning of the generalization process the selection of results other than the best ones is more likely than towards the end. Thus, in the beginning SA chooses operators almost randomly, while in the end it resembles a hill climbing approach. The processing cycle is very similar to the hill climbing approach; only step 4 has to be changed:

(4) with probability *P*: retain another result than the *best result* as new *current data*, decrease *P* and then continue with step 2 with probability 1-*P*: if *cost after* is not lower than *cost before*: return finally the *current data*

The heavy dotted lines in Fig. 9 show which sequence probably can get selected by this search algorithm. Thus for this specific example it is possible to obtain a better but also an inferior result than with hill climbing.

The use of simulated annealing has been demonstrated in map generalization on the example of a displacement algorithm [39] and a displacement algorithm in combination with other generalization algorithms [40]. In the approaches by Ware et al., however, every single building is treated separately in order to decrease the constraint violations. In contrast, the approach presented in this paper applies with every processing step an arbitrary stand-alone generalization operator to an entire building partition. For example, in a particular step all buildings of a partition may be subjected to a simplification operation.

4.2.3 Genetic deep search

In contrast to the two preceding heuristics, so-called genetic algorithms do continue with a subset of the intermediate results instead of only one. [38] demonstrated the use of a genetic algorithm for the displacement of buildings. The genetic deep search strategy shown here starts off by retaining all operator results as intermediate results. With every step the number of results that are retained to proceed, decreases, so that after some steps again only the best result is selected to proceed. This strategy searches not only one possible sequence but a set of sequences. As indicated in Fig. 10 by the heavy lines, in the beginning all three results are retained and then at the next step only the two better results are chosen to proceed. So, from the 27 possible sequences in this example, six are evaluated instead of only one.

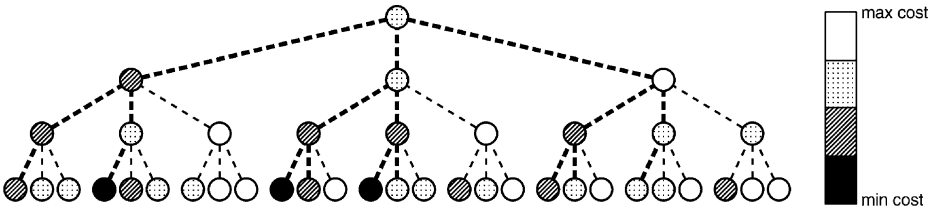

Fig. 9 Simulated annealing, may proceed with a different result than the best one with decreasing probability over time

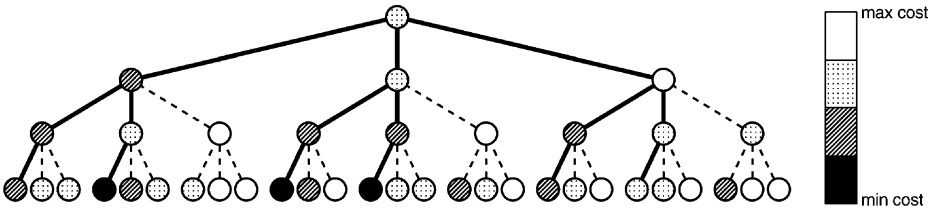

Fig. 10 Genetic deep search, proceeds in parallel with the best and a (decreasing) number of other results

In the experiments with eight operators (see Section 6) all eight results were retained in the first iteration in order to proceed; then four results in the second and third iteration; two in the fourth iteration; and in all following iterations only the best result was selected. This resulted in the evaluation of 64 sequences, each having a maximum length of 20. As with this approach 64 sequences instead of one have to be computed and evaluated this search algorithm is much slower than the two preceding approaches. However, as the experiments showed the likelihood of finding a better minimum is much larger.

5 Implementation as a processing service

The implementation was carried out in a web service framework called WebGen [8], utilizing so-called generalization operator services and generalization support services (described in detail in [25]). As a third type of generalization service, processing services control the sequencing of the different available operator services [8], [25]. Furthermore, they are also using functionalities offered by support services. Most of the currently available services in the WebGen framework are written in Java and thus can be used both as web services over a network or as Java classes on a local computer. In both cases the same interface is used. A client for the WebGen framework is available, among others, for the JUMP Unified Mapping platform [18]. Thus, JUMP is used for accessing and analyzing the processing services. Figure 11 shows the display of a parallel coordinate plot of constraints used for analyzing the results of a processing cycle within JUMP. Results of the various operator and processing services can directly be evaluated and visualized.

The implemented processing services first use a partitioning service to derive the individual building partitions. Then, for every partition (i.e. for every urban or rural block) the available operator services are called and their results evaluated by an evaluation support service. The search then proceeds over the following iterations: repeatedly calling the available operator services and the evaluation support service, according to the search algorithms outlined in the previous section.

5.1 Support services

Two kinds of support services are used by the processing services, namely partitioning and evaluation services. The partitioning services are deriving building partitions based on the trans-hydro graph [37], utilizing the transportation and river networks for the subdivision of the map space into non-overlapping, independent generalization zones (partitions). This partitioning strategy is sufficient for moderate scale changes as in our examples.

Evaluation services are applied for the calculation of constraint violation values for the building partitions before and after generalization. For every constraint a separate

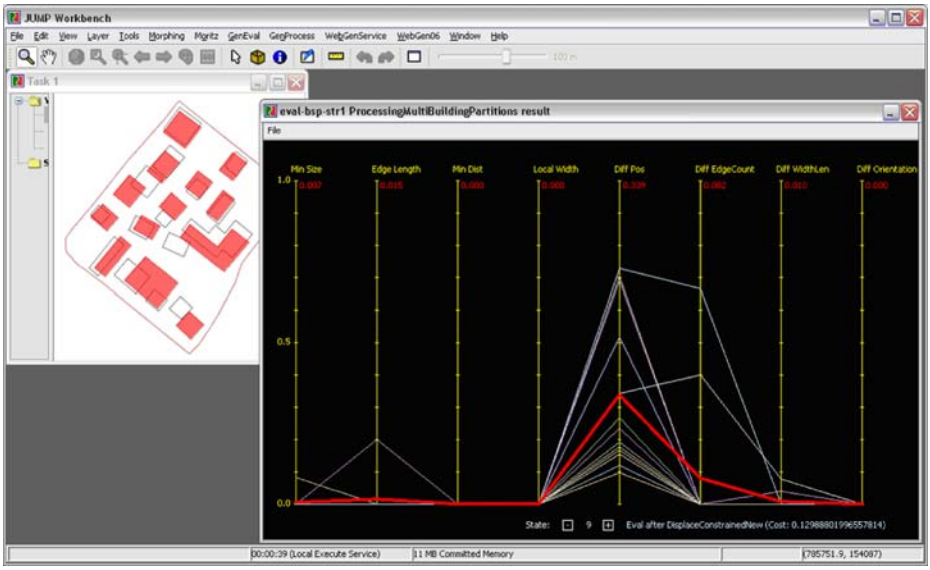

Fig. 11 Screenshot of JUMP client showing constraint violations after generalization within the WebGen framework

evaluation service is available which returns the violation costs. These values are then combined and evaluated by the cost function (see Section 3.3). All evaluation services have the same service interface. Thus new constraints can easily be added by creating a new evaluation service and adding it to the cost function.

5.2 Operator services

The processing strategies presented in this paper have the goal to automatically chain all sorts of independent generalization operators. Every operator that is available as a service within the WebGen framework can be used by the processing strategy. They just have to be registered with the processing service. Thus, new operators can easily be added and due to the constraint-based evaluation new operators are automatically integrated and applied in a sequence if it is appropriate. Operator services are developed to offer the generalization functionalities such as simplification, displacement, typification, or scaling. In Table 1 the generalization operators used in our experiments are briefly described. Sample results can be seen Fig. 4. For downloads and more information on these and other algorithms, see the WebGen server (http://webgen.geo.uzh.ch). Note, however, that the generalization algorithms that have been implemented so far are mainly for demonstration purposes of the web services and the processing services. Hence, they are mainly simple algorithms taken from the literature. Designing optimally efficient and effective generalization algorithms was not the objective of our work.

5.3 Speed-up by parallel processing

The generalization algorithms and workflows shown so far are executed in a sequential, stepwise order. The presented processing strategies use a brute-force search approach by trying out all available algorithms on a particular partition of the map data. Especially for

Table 1 Generalization operator services used in the experiments

Operator name	Description	References
Enlargement	Simply scales buildings that are smaller than the minimum building size about the building centroid	
Simplification	Removes edges that are too short as well as small gaps between not directly connected edges	[35]
Rectification	Simplification and rectified enlargement of buildings smaller than the minimum building size	M. Bader in [1]
Displacement	Aims to achieve sufficient distances between the map features so that they are distinguishable and do not visually coalesce. The algorithm can respect different required minimum distances between buildings and between buildings and roads	[25]; uses the triangulation-based proximity graph by [28]
Typification	Reduces the number of buildings and tries to maintain the arrangement of buildings in a partition. Uses a weighted Delaunay triangulation to replace nearby buildings with a placeholder resembling the more important replaced building	[11]
Aggregation	Aggregates nearby buildings; growing and subsequent shrinking of buffers (dilation and erosion) are used to merge the buildings	
Shrinking	Simply scales down the size of the buildings by a constant factor	
Compression	Compresses and pushes all buildings away from a conflict zone, e.g. a road which is too close, using a rubber-sheeting like approach and reduces the building size accordingly	

the genetic deep search approach huge amounts of processing steps have to be performed. Thus, instead of using an intensive, logically very complex search heuristic the more extensive search approaches try to solve the optimization task by trying out many different possible sequences. Parallel processing of separate partitions as well as the parallel execution of the algorithms on a single partition can improve the performance of these processing strategies substantially.

Parallel computing is no longer limited to specialized hardware and software platforms. The advent of multi-processor computers based on standard PC technology in combination with multi-threaded or distributed programming offers parallel computing functionalities to almost everybody. There exist a great variety of parallel computing approaches. For the processing strategies presented in this paper a multiple instruction and multiple data stream system can be used. This approach subdivides the data and the instructions into independent tasks and computes them in parallel. The advantage is that no concurrency problems occur while accessing the data. This approach can be used both on a computer with multiple processors but also on a cluster of multiple computers linked via a network.

For the parallel processing of generalization tasks both domain and functional decomposition can be used [20]. Domain decomposition divides a job into independent small units. The job in our case consists of all the buildings that have to be generalized. The separation into independent partitions (i.e. the urban blocks) delivers small units of independent data for generalization. Functional decomposition divides a workflow into different tasks as we do by applying the different algorithms onto a building partition in

every iteration. The parallel processing approach presented in this paper executes only completely independent tasks in parallel, which avoids message passing between the separate parallel processes.

Both domain and functional decomposition can be used in a service-based environment using a cluster of multiple instances of the generalization services as shown in Fig. 12. The *Parallel Processing Service* acts as a master process which derives the building partitions and creates a separate instance of the *Processing Service* (1) for every building partition i=1..n. Every service instance is a completely separated process which receives its input data and returns its result data. In every instance of the *Processing Service* a processing strategy, such as the hill climbing search, is executed. During the iterative cycle of the processing strategy with every step all available *Operator Services* j=1..m are applied to the current map state of partition i (2). Thus, every *Operator Service* is applied to the same data. This can again be done in separate service instances. After all *Operator Services* have completed, their results are evaluated in order to retain one of the results. The computation and evaluation of the constraint violations can again be carried out in separate instances of these *Support Services* for the result of each operator j=1..m (3). Thus, at three stages of the processing strategies it is possible to create independent instances that can run in parallel.

Every instance of a service is a process that runs completely independently. The different instances can run on a single computer or also on different computers, due to the web services architecture used. Thus, the load can be distributed. When using the processing services within JUMP on a local computer a separate thread is created for every service instance. These threads can run in parallel if the computer has more than one processor core as it is increasingly the case in standard personal computers. Thus, an immediate speed up is possible. Experiences and results with parallel processing on a multi-processor computer are presented in Section 6.4.

Different partitions can also be processed on different WebGen servers residing on different computers and also the instances of the operator services can be distributed on the different computers (i.e. nodes) of such a parallel processing cluster. In this case a central controller executes the *Parallel Processing Service*. This service calls the *Processing Services* on the different nodes. Currently we use a simple round robin scheduler for this purpose. However, a more advanced scheduler can be imagined which monitors the performance and current load of the various nodes and thus distributes jobs accordingly.

With the distributed parallel processing approach an overhead exists due to the duplication and distribution of the map data to the different services. However, as the

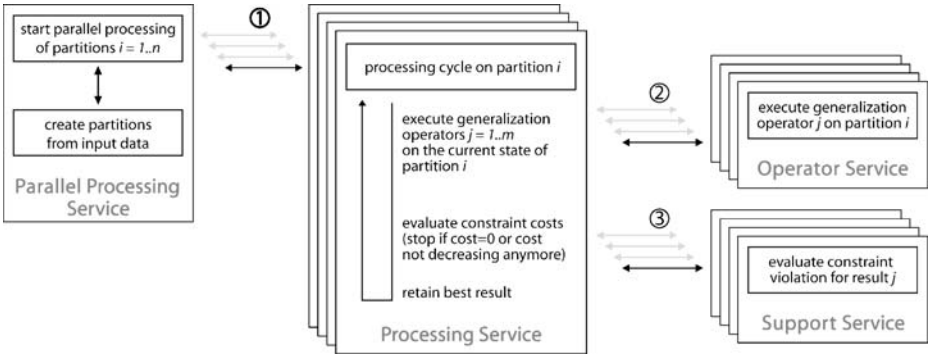

Fig. 12 Parallel processing in the WebGen framework

execution time of the operators usually is much longer than the network transfer time of the relatively small partitions, this overhead only causes a problem when only small numbers of building partitions are processed.

6 Experiments & results

In this section experiments with the processing strategies and their different search algorithms are presented. The WebGen framework with the JUMP client served as the test platform.

6.1 Settings

For the experiments 99 sample partitions (i.e. urban blocks) of Swisstopo VECTOR25 data are used. Every test dataset consists of the buildings in a block and their surrounding roads. The type and density of the building partitions ranges from inner city blocks with many, closely spaced buildings to rural partitions with only few, loosely distributed buildings. The number of buildings ranges from 1 to 80 per partition.

The cost function for the experiments uses the constraints introduced in Section 3. The four constraints MinSize, EdgeLength, MinDist, LocalWidth were used as active constraints with weight 1 since they absolutely should be fulfilled. The remaining four constraints (DiffPos, DiffEdgeCount, DiffWidthLen, and DiffOrientation) were understood as passive constraints intended to preserve certain properties; they were assigned a weight of 0.25. This value was chosen only for these experiments and proved to meet the needs. Using this weighting the cost values range from 5, if all constraints are maximally violated, to 0, if all constraints are perfectly satisfied. In our experiments the average initial cost was around 1.1 and the maximum initial cost around 2.5 with our sample data.

Before starting the processing cycle, four parameters ensuring the legibility of the resulting map are required to initialize the constraints. All other parameters for the generalization operators are derived automatically from these four initial parameters. The four basic parameters include the minimum size of a building, the minimum distance between buildings, the minimum distance between buildings and roads, and the minimum segment length of the building polygons. The parameter settings for the experiments are listed in Table 2.

For the experiments the derivation of a 1:50,000 map from the 1:25,000 VECTOR25 source data was performed. Therefore, we based our settings on the recommendations of the Swiss Society of Cartography [34]. At the 1:50,000 scale the complete traffic network should be maintained. All settlements must be shown as well as isolated individual buildings. Town centers should be maintained and in case of doubt houses may be omitted in favor of the traffic network. According to the legibility recommendations the smallest square on a 1:50,000 map should have an edge length of 0.35 mm on the map and thus of

Table 2 Processing parameters used for initializing the constraints, with a target scale of 1:50,000

Parameter name	Value
Minimum building size	306 m^2
Minimum building distance	10 m
Minimum building-road distance	15 m
Minimum polygon segment length	12.5 m

17.5 m on the ground. This leads to a minimum building size of 306 m^2. The minimum distance between buildings on the map is 0.20 mm (i.e. 10 m on the ground), and the minimum segment length is 0.25 mm on the map (i.e. 12.5 m on the ground). A typical road in rural as well as residential areas has a ground size of 5 m. In the 1:50,000 map it is represented by a line with a thickness of 0.6 mm and thus has a virtual ground width of 30 m. Therefore the minimum distance between buildings and the road axis has to be 15 m.

6.2 Comparison of search algorithms

In order to compare the different search algorithms the above settings were used. The different search strategies with their processing cycles were applied to each of the 99 urban block samples. The most important value is the *average cost* as a measure of the constraint violations after the generalization of all samples (see Section 3.3). Every strategy achieves a reduction of the average cost compared to the initial state (Table 3). The hill climbing (HC) approach and the simulated annealing reduced the average cost by 61% and 63%, respectively, with quite a similar outcome. The genetic deep search (GDS), however, reduced the average cost by almost 78%. This is due to the fact that the genetic approach tests at most 64 possible sequences instead of only one single sequence in the other search algorithms. This can be seen by the *average sequence length* and the *calculated steps*. The two approaches HC and SA have an *average sequence length* as well as *average steps calculated* of 6.7 and 9, respectively, meaning that they calculate exactly the same number of steps (operator executions) for every sample. For GDS, however, the *average sequence length* is 11.6 for obtaining the optimal solution, with 467 steps calculated on average for the evaluation of 64 sequences. Unfortunately, this leads to a much longer processing time. In our test environment (see also Section 6.4) the average execution time for every partition was more than 18 min with GDS, while HC and SA only needed 24 and 32 s, respectively.

This comparison of the different search algorithms shows that the genetic deep search reduces the average cost significantly (more than 40% less) compared to the two other approaches. This gain in cartographic quality, however, can only be reached with a significantly slower execution. The quality gain of simulated annealing compared to hill climbing is only marginal. To achieve this cost reduction of 5%, the execution time grows by 55%, however. This is mainly due to the fact that the average sequence length with SA is 2.2 steps longer than with HC. An explanation for this effect is that a mistaken selection of a fine grained operator (Section 3.4) in an early step of a sequence can be fixed in a later step. Thus, the selection of a sequence that would lead to a high local minimum can be compensated by the execution of other additional operators. The selection of a bad sequence starting with a discrete operator (e.g. aggregation or typification) can, however,

Table 3 Comparison of different optimization methods tested with 99 building partitions

	Initial state	Hill climbing	Simulated annealing	Genetic deep search
Average cost	1.0062	0.3899	0.3695	0.2218
Improvement (from initial)		61.25%	63.28%	77.96%
Improvement (from HC)			5.24%	43.12%
Average processing time		6.73 s	10.47 s	326.21 s
Slowdown (from HC)			55.57%	4,747.10%
Average steps calculated		6.7172	8.9596	467.3737
Average sequence length		6.7172	8.9596	11.6364

not be redone in a later step, leading to the same result as HC. Towards the end of their execution SA and HC behave quite similarly; at that late stage SA cannot better prevent from being trapped in local minimum than HC.

[40] showed an SA approach originally based on a displacement algorithm which is extended with functions to delete buildings or change their size. The optimization steps in their approach have a very fine granularity making only small changes like the displacement of a single building at a time. These small steps do not have instantly large effects if the SA algorithm initially chooses the wrong operation. Mistaken displacements of single buildings can even be redone later. In contrast, the optimization steps of the approach presented in this paper have a quite coarse granularity due to the fact that with every step the complete building partition is treated by the operator. These changes can be much more severe and not reversible at later execution steps. The quasi-random operator choice in the beginning of an SA generalization process can then lead to a completely inferior operator sequence. Thus, applying SA in this setting offers no substantial improvements over HC while having a less good performance.

6.3 Comparison of operator sequences

In order to better understand the behavior of the different search algorithms the positions where the particular operators are executed in a processing sequence can be analyzed. Figure 13 compares this for hill climbing against simulated annealing. Let's first look at the circle sizes only, without distinguishing between HC and SA. Apparently the displacement operator is used much more often than most of the other operators, as indicated by its large circle size. Usually this behavior is caused by minimum distance problems, particularly between buildings and roads. The enlargement operator is executed quite rarely, usually at most once per sample. The shrinking and compression operators both are more likely to get executed later in a sequence because they can sometimes resolve conflicts that are not solvable by displacement. As shown in the examples of Fig. 4 the typification, aggregation, shrinking and compression operators cannot always reduce the cost by themselves but they are needed to reduce the number and size of buildings in order to render the displacement operator successful. Discrete, coarse-grained operators such as simplification and

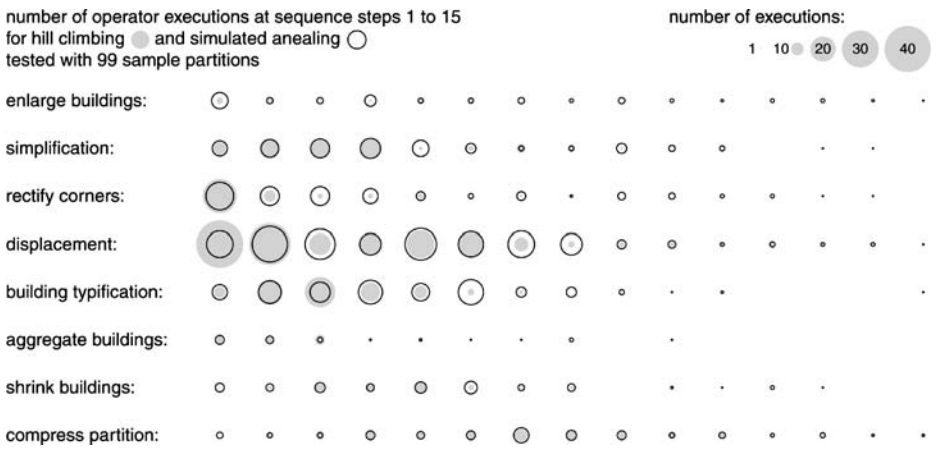

Fig. 13 Comparison of hill climbing and simulated annealing operator sequence positions

rectification are executed relatively often as they usually are required in every partition and they usually reduce cost without creating new conflicts.

Comparing HC and SA it can be observed that the overall number of operator executions does not differ much. The distribution of operator selections in the beginning, however, is more uniform for the SA approach as all operators have a chance of getting executed due to the initially randomized behavior. HC, however, favors the operators which promise a high cost reduction, thus especially the displacement operator and more seldom rectification, simplification and typification. The other operators get executed very rarely in the beginning of a sequence. If SA is used, displacement is not executed as often in the beginning as with HC; however, it is selected somewhat more often in later steps than with HC. This behavior applies also for the typification operator. Apart from these observations the two strategies behave very similarly.

The comparison of the operator positions for the hill climbing and the genetic deep search approaches is shown in Fig. 14. The first striking difference is that GDS uses many more executions than HC. This is clearly true as evidenced in Table 3 by the average sequence length. GDS has an average length of 11.6 against 6.7 for HC. Thus, almost twice as many operator executions take place in an average sequence.

When comparing the two approaches HC and GDS it can be seen that especially the shrinking but also the typification operators are used much more often with GDS. With HC these operators are not very popular because they are initially not reducing the cost very much. GDS, however, evaluates also these sequences that start off with such a less favorable operator. Thus a sequence can be found that leads to a better global minimum than with HC.

6.4 Parallel processing performance

As already introduced in Section 5.3 a parallel processing approach was used in order to speed up the process. In a two stage approach during the iterative processing cycle, for every partition the executions and evaluations of the different operators were computed in separate, parallel threads. The execution of these independent processing cycles again was parallelized so that more than one partition was treated in parallel. This strategy, outlined in

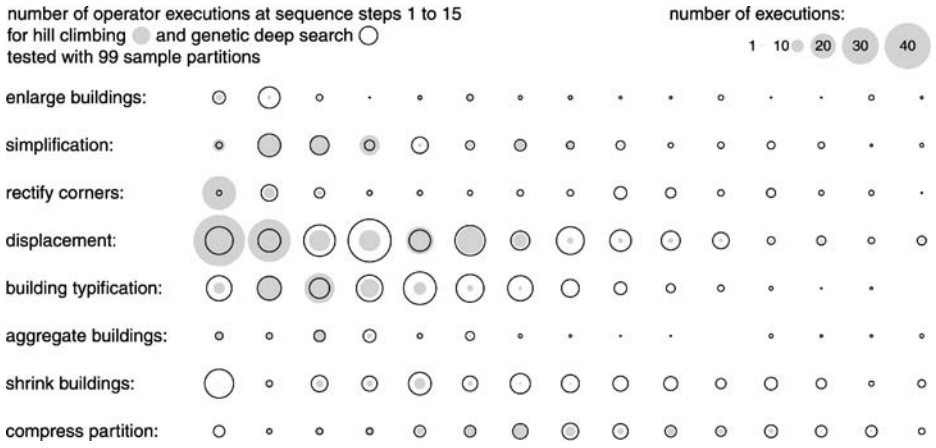

Fig. 14 Comparison of hill climbing and genetic deep search operator sequence positions

Fig. 12, helped to overcome the problem that every iteration cycle is only as fast as the slowest operator, since the evaluation and selection can only be carried out when all results are available. The experiments were conducted on a server equipped with four dual-core processors running at 2.19 GHz clock speed under the Windows 2003 Server operating system. Figure 15 shows the CPU usage history for the processing of a dataset with nine building partitions using the hill climbing approach. With sequential execution the first two processors (with one CPU used for the scheduler) get used, but as the program is sequential, the system load never exceeds the 12.5% of a single processor. With parallel execution all eight CPUs get used and thus the complete system load is approximately 60% for this example. The complete processing time decreases from 33 s (sequential) to only 17 s (parallel).

Thus, already with nine sample partitions a significant speed up could be achieved. The processing time has been reduced to 51% compared with the sequential approach. The system load fluctuates due to interferences between the Windows scheduling and the thread scheduling of the Java Virtual Machine. Working with larger numbers of sample partitions the performance gain can be increased to a constantly higher system load than the 60% in the above example. In performance tests with all 99 sample partitions the system load reached the 100% particularly during the initial phase when still many small partitions were processed in parallel and only later dropped to a low of 13–20% towards the end when only few large and complex partitions remained to be finished. An intelligent scheduler knowing about this problem could increase the performance by trying to mix the processing of complex and simple partitions. The processing of 99 samples (same as used for Table 3) with the hill climbing approach took 2,799 s with sequential and 667 s with parallel execution, yielding a reduction by 76%. The average overall system load was approximately 51%.

7 Discussion

7.1 Global vs. local constraint satisfaction

The processing strategies used in this paper are all based on the ability to describe the desired cartographical quality sufficiently by means of constraints. Currently the constraints are only evaluated globally for each partition, that is, the approach is controlled by the aggregate satisfaction of the constraints of all features in one partition (i.e. by the cost function described in Section 3.3). Remedies for specific constraint violations of one

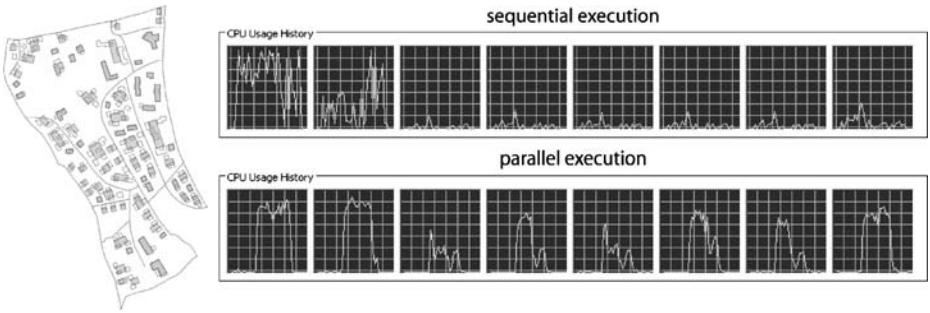

Fig. 15 Parallel processing CPU usage: 33 s (*sequential*), 17 s (*parallel*)

particular map feature are therefore in the responsibility of the individual operators as they are usually trying to solve the most important constraint violations first.

7.2 Approximating deep search

Among the three tested heuristics the *genetic deep search* approach provided the reference results. This approach tries to approximate a complete exhaustive search and thus to test as many different sequences as possible. A real exhaustive search is clearly not feasible due to the overwhelming complexity. Another combination of a search heuristic and a *deep search* could be the concatenated use of deep searching with low depth. For example, compute the full search space with depth 3, then take the best result and proceed again by computing this limited search space for this intermediate best result. Such approaches would probably benefit significantly from the parallelization.

7.3 Combining automated processing with pre-defined workflows

The approaches presented in this paper only looked at the generalization of buildings constrained by their surrounding roads. In a real scenario these roads as well as all other map features must be generalized accordingly. Following the recommendations of [34] for the scale transition from 1:25,000 to 1:50,000 no real road generalization is necessary as most of the roads need not change and only few are eliminated (e.g. dead ends). For a larger scale change (e.g. from 1:25,000 to 1:100,000) more substantial road generalization including elimination, simplification and displacement must be applied. Therefore a combined semi-automatic procedure using predefined workflows in combination with constraint-based processing strategies could be envisioned. A workflow modeling system [26] could then trigger different static and automated generalization services:

- road elimination by road class (e.g. constrained by a connectivity graph)
- partitioning (buildings inside a block are selected, preserving topology)
- road simplification (including road displacement if possible)
- building generalization of the partitions (constrained by the pre-generalized roads)
- road displacement (if needed and possible)

The two last steps possibly could be iterated in order to resolve major conflicts between roads and buildings. The workflows themselves can again be seen as one generalization operator. Such an opaque service would hide the complex workflow behind a simple interface. Even the use of such an operator in another workflow or automated process is imaginable.

7.4 Optimization through machine learning and collaborative filtering

Features or groups of map features can be placed inside a constraint space [9] depending on their cartographic properties. The distance of the features from the origin of this constraint space is a measure of cartographic conflicts. Features at the origin are not violating any cartographic constraints. The constraints allow the identification of similar cartographic situations for the features and partitions, which will be generalized with the same generalization operator. Thus, for example a situation with many minimum distance and minimum building size violations would indicate the use of a typification first, in order to reduce the number of buildings, followed by a displacement.

The method suggested by [10] uses the constraint space for the prediction of generalization operator sequences. This method exploits a knowledge base to predict the operator sequences and proceeds in two phases, a training phase and an application phase. In the training phase, the knowledge base is populated with information about successfully generalized features (described by their position in the constraint space), and the corresponding operator sequences and parameter settings that lead to the successful result. In the application phase the constraint patterns of a given map object are matched against the trained constraint patterns to retrieve similar operator sequences. Thus, the most promising operators with their associated parameter settings are applied to the map object. A collaborative filtering technique [21] guaranties a fast retrieval of operator sequences from a large knowledge base.

As the results reported in Section 6.2 show, the *genetic deep search* approach may be prohibitively slow when the execution time plays a role. However, as it delivers significantly better results than the other approaches it could serve as a search strategy for the training phase of the knowledge base leading to the storage of better sequences than can be found with other search algorithms, such as hill climbing and simulated annealing.

7.5 Extensibility and portability of the optimization approach

The presented services architecture (as described in Section 5) allows easy extension by new operators and constraints. New operators, available as operator services, can just be added to the list of operators in the processing service that is performing the optimization strategy. The new operators are then automatically applied and evaluated. The same extensibility applies also for the constraints. Each constraint is calculated by a separate support service. Thus, new constraints only have to be added to the cost function in the processing service in order to be taken into account while evaluating a partition.

The application of the presented approach to other feature classes than buildings and roads would require other, more advanced, ways to divide the map space. The strategy of applying and evaluating several operators as separate services operating in parallel, however, can easily be employed in other application scenarios.

8 Conclusions

Constraint-based combinatorial optimization techniques can be used for controlling the selection and chaining of different generalization operators of varying sophistication and task granularity. The operators as well as the supporting functions and the iterative processing methods are implemented within the WebGen generalization web services environment that was originally developed with the aim of building a common research platform [8] and later extended by additional functionality [25]. The operators used are true stand-alone services and have no direct interdependencies; the web service interfaces are the only communication and data transfer utility. In this paper, we have now demonstrated how the versatility of a service-based architecture can be exploited in generating automated generalization processes, in a way that is not possible with more traditional computing paradigms. In particular, the service-based approach allows to build modular generalization processes that can be flexibly extended by introducing additional generalization operators or supporting facilities, that can be coupled to arbitrary optimization strategies, and that can be parallelized in order to speed up computationally demanding processes.

Different optimization algorithms including *hill climbing*, *simulated annealing* and *genetic deep search* can be used and have been shown to produce different results in terms of the amount of conflict reduction and processing performance. For the experiments partitions of buildings separated by a trans-hydro-graph were generalized automatically. The *genetic deep search* evaluates a range of possible operator sequences, as opposed to only one as in the other optimization algorithms. Thus, the amount of conflict reduction is significantly higher, but the processing time needed inhibits the use with larger datasets or in near real-time environments. It is interesting to note that with the presented processing strategy the *simulated annealing* approach only optimizes slightly better than the *hill climbing* approach while needing significantly more processing time. This finding is in contrast with the results of [40]; the main reason for this difference being the different granularity of the operators implemented (see Section 6.2). Parallel processing proved to be useful for increasing the processing performance. As a future step the combination of the *genetic deep search* approach with machine learning techniques will be investigated. Initial work in this direction has been reported in 0.

Acknowledgements This research was partially funded by the Swiss National Science Foundation (grant 20-101798, project DEGEN). Thanks go to Ingo Petzold, Stefan Steiniger and three anonymous reviewers for helpful comments.

References

1. AGENT Consortium. “Deliverable D1—Specification of Basic Algorithms”. University of Zurich: Department of Geography, 1999.
2. M. Bader, M. Barrault, and R. Weibel. “Building displacement over a Ductile Truss,” International Journal of Geographical Information Science, Vol. 19(8-9):915–936, 2005.
3. S. Bard. “Quality assessment of cartographic generalization,” Transactions in GIS, Vol. 8(1):63–81, 2004.
4. M. Barrault, N. Regnauld, C. Duchêne, K. Haire, C. Baeijs, Y. Demazeau, P. Hardy, W. Mackaness, A. Ruas, and R. Weibel. “Integrating multi-agent, object-oriented and algorithmic techniques for improved automated map generalization,” in Proceedings of the 20th International Cartographic Conference, Beijing, China, pp. 2110–2116, 2001.
5. K. Beard. “Multiple representations from a detailed database: a scheme for automated generalization”, Ph.D. thesis, University of Wisconsin, Madison, 1988.
6. K. Beard. “Constraints on rule formation,” in B. Buttenfield and R. McMaster (Eds.), Map Generalization: Making Rules for Knowledge Representation. Longman: London, 121–135, 1991.
7. D. Burghardt and S. Meier. “Cartographic displacement using the snakes concept,” in W. Foerstner and L. Pluemer (Eds.), Semantic Modeling for the Acquisition of Topographic Information from Images and Maps. Birkhaeuser: Basel, 59–71, 1997.
8. D. Burghardt, M. Neun, and R. Weibel. “Generalization services on the web—a classification and an initial prototype implementation,” Cartography and Geographic Information Science, Vol. 32(4):257–268, 2005.
9. D. Burghardt and S. Steiniger. “Usage of principal component analysis in the process of automated generalisation,” in Proceedings of 22nd International Cartographic Conference La Coruña, Spain, 2005.
10. D. Burghardt and M. Neun. “Automated sequencing of generalisation services based on collaborative filtering,” in M. Raubal, H.J. Miller, A.U. Frank, and M. Goodchild (Eds.), Geographic information science, 4th International Conference on Geographical Information Science (GIScience), IfGIprints 28, pp 41–46, 2006.
11. D. Burghardt and A. Cecconi. “Mesh simplification for building typification,” International Journal of Geographical Information Science, Vol. 21(3):283–298, 2007.
12. D. Burghardt, S. Schmid, and J. Stoter. “Investigatios on cartographic constraint formalisation,” in 10th ICA Workshop on Generalization and Multiple Representation, Moscow, 2007.
13. B. Buttenfield and R. McMaster. Map Generalization: Making Rules for Knowledge Representation. London: Longman, 1991.

14. P. Gray, L. Painton, C. Phillips, M. Trahan, J. Wagner. "A survey of global optimization methods," Technical report, http://www.cs.sandia.gov/opt/survey/main.html (accessed 02/2007), 1997.
15. L. Harrie. "The constraint method for solving spatial conflicts in cartographic generalization," Cartography and Geographic Information Science, Vol. 26(1):55–69, 2000.
16. L. Harrie and R. Weibel. "Modelling the overall process of generalisation," in A. Ruas, W.A. Mackaness, and T. Kilpeläinen (Eds.), Generalisation of Geographic Information: Cartographic Modelling and Applications. Elsevier: Amsterdam, 67–87, 2007.
17. G. Heuvelink and E. Pebesma. "Spatial aggregation and soil process modeling," Geoderma, Vol. 89(1-2):47–65, 1999 April.
18. JUMP. "The JUMP Unified Mapping Platform," http://www.jump-project.org, 2007.
19. S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. "Optimization by simulated annealing," Science, Vol. 220 (4598):671–680, 1983.
20. G. Langran. "Generalization and parallel computation," in B. Buttenfield and R. McMaster (Eds.), Map Generalization: Making Rules for Knowledge Representation. Longman: London, 204–216, 1991.
21. G. Linden, B. Smith, and J. York. "Amazon.com recommendations. Item-to-item collaborative filtering," IEEE Internet Computing, Vol. 7:76–80, 2003.
22. R. McMaster and S. Shea. Generalization in Digital Cartography. Association of American Geographers: Washington, USA, 1992.
23. S. Mustière. "Cartographic generalization of roads in a local and adaptive approach: a knowledge acquisition problem," International Journal of Geographical Information Science, Vol. 19(8-9):937–955, 2005.
24. S. Mustière, J.-D. Zucker, L. Saitta. "An abstraction-based machine learning approach to cartographic generalization," in 9th International Symposium on Spatial Data Handling (SDH 2000), Beijing, China, 50–63, 2000.
25. M. Neun, D. Burghardt, and R. Weibel. "Web service approaches for providing enriched data structures to generalisation operators," International Journal of Geographic Information Science, Vol. 22(2):133–165, 2008.
26. I. Petzold, D. Burghardt, and M. Bobzien. "Workflow management and generalisation services," in 9th ICA Workshop on Generalization and Multiple Representation, Portland, 2006.
27. N. Regnauld. "Constraint based mechanism to achieve automatic generalization using agent model," in Proceedings of the GIS Research UK (GISRUK 2001), pp. 329–332, University of Glamorgan, 2001.
28. N. Regnauld. "Spatial Structures to Support Automatic Generalisation," in Proceedings of the XXII International Cartographic Conference, A Coruña, Spain, 2005.
29. D. Richardson and J.-C. Muller. "Rule selection for small-scale map generalization", in B. Buttenfield and R. McMaster (Eds.), Map Generalization: Making Rules for Knowledge Representation. Longman: London, 136–149, 1991.
30. A. Ruas and C. Plazanet. "Strategies for automated generalization," in Proceedings of the 7th International Symposium on Spatial Data Handling (SDH 1996), pp. 319–336, Delft, the Netherlands, 1996.
31. A. Ruas. "Modèle de généralisation de données géographiques à base de contraintes et d'autonomie," Ph.D. thesis, IGN France and Université de Marne La Vallée, 1999.
32. A. Ruas and C. Duchêne. "A prototype generalisation system based on the multi-agent system paradigm," in A. Ruas, W.A. Mackaness, and T. Kilpeläinen (Eds.), Generalisation of Geographic Information: Cartographic Modelling and Applications. Elsevier: Amsterdam, 269–284, 2007.
33. M. Sester. "Generalization based on least-squares adjustment," International Archives of Photogrammetry and Remote Sensing, Vol. XXXIII:931–938, 2000 Part B4, Amsterdam.
34. Swiss Society of Cartography. Topographic Maps—Map Graphics and Generalisation. Swiss Society of Cartography: Wabern, Switzerland, 1995.
35. W. Staufenbiel. "Zur Automation der Generalisierung topographischer Karten mit besonderer Berücksichtigung großmaßstäbiger Gebäudedarstellungen," Institute of Cartography and Geoinformatics, University of Hannover, No. 51, 1973.
36. S. Steiniger and R. Weibel. "Relations among map objects in cartographic generalization," Cartography and Geographic Information Science (CaGIS), Vol. 34(3):175–197, 2007.
37. S. Timpf. "Hierarchical structures in map series," Ph.D. thesis, Technical University Vienna, 1998.
38. I.D. Wilson, J.M. Ware, and J.A. Ware. "A genetic algorithm approach to cartographic map generalization," Computers in Industry, Vol. 52(3):291–304, 2003.
39. J.M. Ware and C.B. Jones. "Conflict reduction in map generalization using iterative improvement," GeoInformatica, Vol. 2(4):383–407, 1998.

40. J.M. Ware, C.B. Jones, and N. Thomas. "Automated map generalization with multiple operators: a simulated annealing approach," International Journal of Geographical Information Science, Vol. 17 (8):743–769, 2003.
41. R. Weibel, S. Keller, and T. Reichenbacher. "Overcoming the knowledge acquisition bottleneck in map generalization: the role of interactive systems and computational intelligence," in Proceedings of 2nd International Configuration on Spatial Information Theory (COSIT 95), pp. 139–156, 1995.
42. R. Weibel and G. Dutton. "Constraint-based automated map generalization," in Proceedings of the 8th International Symposium on Spatial Data Handling, pp. 214–224, 1998.

Moritz Neun obtained a Master of Science in Computer Science with a minor in geography from the University of Fribourg (Switzerland) in 2004 and a Ph.D. degree in Geography with a specialization in GIScience from the University of Zurich in 2007. He is research assistant at the "Geographic Information Systems" division at the University of Zurich. His current research interests are the control of map generalization, advanced data structures for map generalization and geographic web services for map generalization.

Dirk Burghardt received his Ph.D. in geosciences from Dresden University in 2000, on the topic of automated generalization. Later he worked as a developer and product manager for a cartographic production company. Currently he is research associate at the Department of Geography at the University of Zurich. His research interests include cartographic visualization, mobile information systems and automated cartographic generalization.

Robert Weibel is Professor of Geographical Information Science at the Department of Geography, University of Zurich. He obtained an MSc and Ph.D. degree in Geography with a specialization in GIScience from the University of Zurich in 1985 and 1989, respectively. Before starting his academic career he was a software engineer in the GIS industry. He has served as chair of the Commission on Map Generalization of the International Cartographic Association (ICA, 1992–2003) as well as on various journal editorial boards and program committees of scientific meetings and conferences in GIScience. His current research interests are in map generalization and multiple representation, location-based services, and spatio–temporal analysis of moving point objects.