# OPTIMAL SIMPLIFICATION OF BUILDING GROUND PLANS


Jan-Henrik Haunert[a] and Alexander Wolff[b]

[a]Institute of Cartography and Geoinformatics, Leibniz Universität Hannover, Germany. – jan.haunert@ikg.uni-hannover.de
[b]Faculteit Wiskunde en Informatica, Technische Universiteit Eindhoven, The Netherlands. – http://www.win.tue.nl/˜awolff



**Commission II/3**

**KEY WORDS:** Cartography, Scale, Generalization, Optimization, Buildings, Simplification, NP-hard, Integer programming



**ABSTRACT:**

This paper presents an optimization approach to simplify building ground plans given by two-dimensional polygons. We define a simplified building as a subsequence of the original building edges; its vertices are defined by intersections of consecutive (and possibly extended) edges in the selected sequence. Our aim is to minimize the number of edges subject to a user-defined error tolerance. We prove that this problem is NP-hard when requiring that the output consists of non-intersecting simple polygons. Thus we cannot hope for an efficient, exact algorithm. Instead, we propose a heuristic and an integer programming formulation that can be used to solve the problem with existing optimization software. We discuss results of our algorithms and how to incorporate more sophisticated objective functions into our model.


## 1 INTRODUCTION

The simplification of building outlines is a well-known problem in cartography: in order to produce readable maps at a smaller scale and to provide data sets at an appropriate level of abstraction, some details from a given large-scale representation of the building need to be omitted. In fact the problem is similar to the classical line simplification problem, but, as buildings are highly regular man-made structures, special characteristics need to be considered. Because of this, solutions for both problems have been developed, on the whole, independently. In this paper, we present a new optimization approach to the building simplification problem, which is inspired by a commonly-used approach to line simplification. Our motivation to approach the problem by optimization is, of course, to obtain generalization results of higher quality compared to other methods. However, there is a second reason for our approach. With our method it is possible to apply different optimization objectives and constraints. We assume that comparing the results will help to better understand the criteria that make up a well-generalized building. This will be useful for quality assessment – a problem that is considered of high relevance in the generalization literature (Bard, 2004).

A classical approach to building simplification is based on defining rules that are successively applied to the polygonal outline of the building (Staufenbiel, 1973). Kada and Luo (2006) define parts of simplified buildings as intersections of half-planes, which are defined by lines that approximate parts of the original building outline. Mayer (1998) applies mathematical morphology to simplify a building. Sester (2005) suggests a two-step procedure: firstly, details are removed by application of rules; secondly, the simplified building ground plan is optimally adjusted, for example, edges are moved to increase the size of a building part. The second step is based on least squares adjustment. However, the first step, i.e., the decision about which details to select, has not been approached by optimization yet.

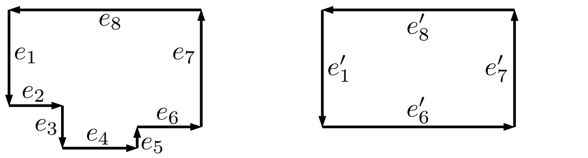
The common line simplification approach is to select a subsequence of vertices from the given polygonal line. Often a simplified line is considered feasible if it satisfies the *bandwidth criterion*, that is, if the original line is within an $\varepsilon$-buffer of the simplified line. Subject to this constraint, Deveau (1985) suggested to minimize the number of vertices. This basic optimization approach to line simplification has been implemented in commercial GIS software, e.g., FME® of Safe Software. To find a simplification with a minimum number of vertices, shortest-path algorithms can be applied to an appropriately defined graph of shortcuts. Campbell and Cromley (1991) define a general model that allows to incorporate different optimization criteria. Also constraints can be defined. The problem becomes more involved, but solutions have been found for preservation of angles (Chen et al., 2005), distances (Gudmundsson et al., 2007), areas (Bose et al., 2006), and topological relations (de Berg et al., 1998).

This paper presents a new basic approach to building simplification that is similar to the general line simplification approach. The main difference is that we define the simplified outline of a building by selecting a subsequence of the original edges and not a subsequence of the vertices. New vertices are introduced at intersections of consecutive edges in the selected sequence. With this approach we keep the edge slopes fixed and so give consideration to shape regularities. For example, if the original building is rectilinear, the simplified building will automatically be rectilinear, too. The optimization objective that we apply is very basic, i.e., we minimize the number of segments in the output subject to a given distance tolerance. Additionally, we will discuss in detail how intersections of polygon edges can be avoided.

The structure of the paper is as follows. We start with a formal definition of the problem (Sect. 2) and prove its NP-hardness (Sect. 3). Section 4 develops a basic method, which can be used to efficiently solve a relaxed problem. Due to the NP-hardness of the original problem we turn to integer programming (Sect. 5) and a heuristic approach (Sect. 6). Finally, we present results of our methods (Sect. 7).

## 2 PROBLEM DEFINITION

We consider a building as a counterclockwise oriented sequence $P = (e_1, e_2, \ldots, e_n)$ of edges of a simple polygon. Our aim is to select edges that catch the main characteristics of the building depending on a user-defined error tolerance. In order to "glue together" consecutive edges $e_k$ and $e_l$ (with $k < l$) in the output, we define the *L-shape* of the edge pair $(e_k, e_l)$ as the union of two rays, both starting at the intersection point of the straight




(a) original building (b) result of applying shortcut $(e_1, e_6)$

Figure 1: Building simplification based on shortcuts.

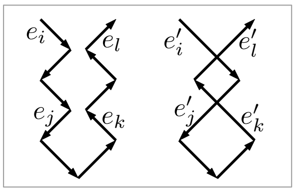



Figure 2: Applying shortcuts $(e_i, e_j)$ and $(e_k, e_l)$ results in a non-simple polygon, though each of them is feasible alone.

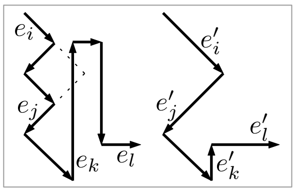



Figure 3: Applying shortcuts $(e_i, e_j)$ and $(e_k, e_l)$ results in a simple polygon, though $(e_i, e_j)$ is not feasible alone.

lines supporting $e_k$ and $e_l$. The first ray runs in direction opposite to $e_k$, the second ray in direction of $e_l$. Since a pair of consecutive edges $(e_k, e_l)$ in the output means to omit input edges $e_{k+1}, e_{k+2}, \ldots, e_{l-1}$, we refer to $(e_k, e_l)$ as a *shortcut*. The general idea is shown in Fig. 1 where the pair $(e_1, e_6)$ shortcuts four edges. The resulting generalized building keeps the characteristic shape of the original building.

Similar to the bandwidth criterion in line simplification we insist that the distance between the L-shape of a shortcut and the omitted edges is small, that is, bounded by a user-defined threshold $\varepsilon > 0$. To express this precisely, we define the *$\varepsilon$-buffer* of a set $A$ of points in the plane as the union of all closed radius-$\varepsilon$ disks centered at points in $A$. For example, Fig. 5(a) depicts the $\varepsilon$-buffer (shaded) of the L-shape defined by the shortcut $(e_k, e_l)$. We can now give a formal definition of our problem.

**Problem** (BUILDINGSIMPLIFICATION)**.** *Given a simple polygon $P = (e_1, e_2, \ldots, e_n)$ as a counterclockwise oriented sequence of directed edges and a distance threshold $\varepsilon > 0$, find a polygon $P' = (e'_{i_1}, e'_{i_2}, \ldots, e'_{i_m})$ with $1 \leq i_1 < i_2 < \ldots < i_m \leq n$ such that $P'$ has the fewest edges among all polygons that fulfill the following requirements:*

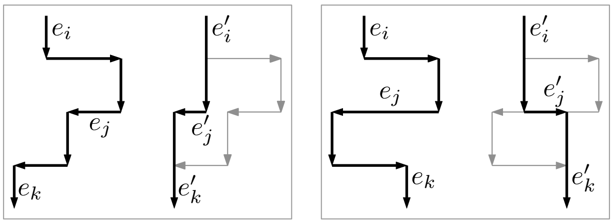
*(R1) $P'$ is simple,*
*(R2) for $j = 1, \ldots, m$ it holds that $e_{i_j}$ and $e'_{i_j}$*
- *(a) intersect and*
- *(b) have the same (directed) supporting line,*

*(R3) for each pair of consecutive edges $(e'_k, e'_l)$ in $P'$,*
- *(a) the sequence $(e_{k+1}, e_{k+2}, \ldots, e_{l-1})$ is within an $\varepsilon$-buffer of the L-shape defined by $(e_k, e_l)$ and*
- *(b) the L-shape defined by $(e_k, e_l)$ enters and leaves the $\varepsilon$-buffer of the sequence $(e_{k+1}, e_{k+2}, \ldots, e_{l-1})$ exactly once.*

We call any simplification $P'$ of $P$ that fulfills requirements (R1)–(R3) *($\varepsilon$-) feasible*. We now motivate the above requirements.

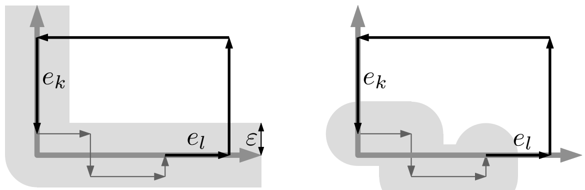
Figure 2 shows an example of two potentially feasible shortcuts. Both imply extensions of edges, and so, applying them together results in intersections, i.e., a violation of requirement (R1). Applying shortcut $(e_i, e_j)$ in Fig. 3 would result in an intersection with edge $e_k$. However, if shortcut $(e_k, e_l)$ is applied simultaneously, the result is simple.

Figure 4 shows violations of requirement (R2). In Fig. 4(a), the edge $e'_j$ corresponding to $e_j$ reappears at a new location. In this case, the simplified outline can hardly be understood as a subsequence of the original edges. We therefore forbid such cases. Nevertheless, we allow to trim or extend an edge at both ends. Figure 4(b) shows a subsequence $(e_i, e_j, e_k)$, which is not feasible since $e_j$ needs to be turned to generate a closed outline. This example, just as Fig. 2, shows that, in order to test the feasibility of a subsequence, it does not suffice to independently test the feasibility of all applied shortcuts: each of the shortcuts $(e_i, e_j)$ and $(e_j, e_k)$ is feasible alone.




(a) requirement (R2a) violated since $e_j \cap e'_j = \emptyset$ (b) requirement (R2b) violated since $e_j$ and $e'_j$ have different directions

Figure 4: Non-feasibility of shortcuts in special cases.




(a) requirement (R3a) is satisfied (b) requirement (R3b) is satisfied

Figure 5: Feasibility of shortcut $(e_k, e_l)$. $\varepsilon$-Buffers are shaded. The bold gray arrows indicate the L-shape defined by $(e_k, e_l)$.

The shortcut $(e_k, e_l)$ in Fig. 5 is feasible according to requirement (R3), to which we refer as bandwidth criterion. Though our criterion consists of two parts, it is similar to the bandwidth criterion in line simplification. The first part (R3a) means that the original line must lie in the $\varepsilon$-buffer of the simplified line. The second part (R3b) means that the simplified line must lie in the $\varepsilon$-buffer of the original line. When reducing a line to a subsequence of *vertices*, the first part implies the second one. However, Fig. 6 shows that this is not true with our definition of shortcuts: without requirement (R3b) we can create new building vertices far from the original shape, while requirement (R3a) is satisfied.

Polygons with holes—for example, buildings with inner yards—are represented by multiple sequences of edges. Given such an input, we reduce each sequence to a subsequence subject to requirements (R1)–(R3). The resulting polygon must be simple; we aim at minimizing the total number of edges.

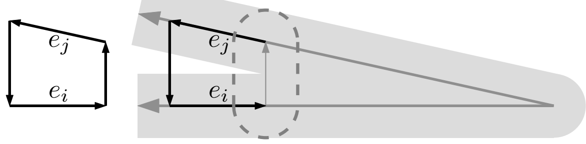



Figure 6: Shortcut $(e_i, e_j)$ satisfies requirement (R3a) (shaded buffer), but does not satisfy requirement (R3b) (dashed buffer).

## 3 COMPLEXITY

We now investigate the computational complexity of BUILDING-SIMPLIFICATION. We prove that the problem is NP-hard un-

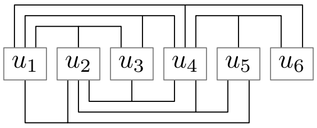



Figure 7: Layout of the variable-clause graph $G_{\mathrm{vc}}$.

der two conditions, namely that (a) the output consists of simple polygons (requirement (R1)) and (b) the input can consist of multiple disjoint polygons or a single polygon with multiple holes. The NP-hardness justifies that we consider non-efficient exact or efficient non-exact solutions in the remainder of this paper.

Starting point of our proof is the decision version of BUILDINGSIMPLIFICATION: given a simple polygon $P$, a threshold $\varepsilon > 0$, and an integer $K > 0$, does $P$ have an $\varepsilon$-feasible simplification with at most $K$ edges? Depending on the answer we say that $(P, \varepsilon, K)$ is a *yes*- or a *no*-instance. As usual, we prove hardness by specifying a so-called *polynomial reduction* from some other (decision) problem $X$ that is known to be NP-hard. For each instance $I$ of problem $X$ we construct (in polynomial time) an instance $I'$ of our problem such that $I'$ is a yes-instance if and only if $I$ is a yes-instance. In other words, solving our problem is at least as hard as solving $X$. As $X$ is NP-hard, our problem is NP-hard, too.

As problem $X$ we use PLANAR3SAT, a variant of the well-known problem 3SAT. An instance of 3SAT is given by a set $U$ of Boolean variables and a set $C$ of *clauses* over $U$. A clause over $U$ is a set of *literals*, each being a variable $u \in U$ or its negation $\bar{u}$, that is, $\bar{u}$ is true if and only if $u$ is false. In 3SAT each clause contains *three* literals. The task is to decide whether it is possible to assign Boolean values (*true* or *false*) to the variables in $U$ such that each clause in $C$ contains at least one true literal.

To define PLANAR3SAT, we define the (bipartite) *variable-clause graph* $G_{\mathrm{vc}}$ to contain a node for each variable in $U$ and for each clause in $C$. The graph contains an edge $\{u, c\}$ for a variable $u \in U$ and a clause $c \in C$ if and only if $u \in c$ or $\bar{u} \in c$. The problem PLANAR3SAT is the special version of 3SAT where $G_{\mathrm{vc}}$ is planar. PLANAR3SAT is known to be NP-hard (Lichtenstein, 1982). Note that $G_{\mathrm{vc}}$ can be laid out (in polynomial time) such that variables correspond to disjoint vertically aligned boxes and clauses correspond to non-crossing three-legged "combs" that are attached to the boxes from above or below (Knuth and Raghunathan, 1992), see the example in Fig. 7.

**Theorem.** BUILDINGSIMPLIFICATION *is NP-hard, that is, given a polygon $P$ (with holes), a threshold $\varepsilon > 0$, and an integer $K > 0$, it is NP-hard to decide whether $P$ has an $\varepsilon$-feasible simplification with at most $K$ edges.*

*Proof.* As noted above, our proof is by reduction from PLANAR3SAT. Our task is to construct an instance $(P, \varepsilon, K)$ of the decision version of BUILDINGSIMPLIFICATION for a given instance $(U, C)$ of PLANAR3SAT such that $P$ has a simplification with at most $K$ edges if and only if $(U, C)$ has a fulfilling truth assignment. We set $\varepsilon = 2$ and specify $K$ later. Note that simplifying a set of disjoint polygons is the same as simplifying a large polygon with many holes. Here we view $P$ as a set of polygons.

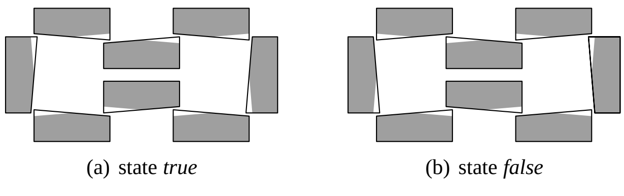
We specify $P$ by assembling a set of *gadgets*; a variable gadget for each variable $u \in U$ and a clause gadget for each clause $c \in C$. The variable gadget is shown in Fig. 8; its exact dimensions are defined in Fig. 9. The variable gadget is a ring of pentagons, each being the union of a rectangle and a flat, isosceles triangle. Only the two simplifications in Fig. 8 are optimal for $\varepsilon = 2$. Other simplifications of equal cost would violate the simplicity requirement. We refer to the two optimal solutions as *true state* and *false state* of the variable gadget, which correspond to values *true* and *false* of the corresponding variable in $U$. Figure 10 defines the clause gadget, which consists of a 14-gon that has five different optimal (10-gonal) simplifications for $\varepsilon = 2$.

Next we connect variable and clause gadgets. We define connectors that can transmit some kind of pressure from a variable to a clause gadget. The connectors are such that there is an optimal simplification of the clause gadget if and only if there no pressure from at least one of the three incident variable gadgets, see Figure 11. A connector consists of the same pentagons as the variable gadget. Figure 11(a) shows a connector that connects a variable $u$ to a clause $c$. The connector transmits pressure if $u$ is *false*. In that case there is only one feasible simplification of the pentagons in the connector. This setting is used if $u$ appears non-negated in the clause. The connector in Fig. 11(b) transmits pressure if $u$ is *true*. This setting is used if $\bar{u}$ is contained in $c$.

If a variable is contained in several clauses we simply increase the number of pentagons in the ring that forms the variable gadget. This yields more slots for connectors. Figure 12 shows how the connectors from three variable gadgets meet at a clause gadget. In the example, each connector transmits pressure from the variable. Observe that in this situation it is not possible to simplify the polygon of the clause gadget; none of the simplifications in Fig. 10 is feasible. On the other hand, if there is no pressure from variable $u$ (resp. variable $v$, variable $w$) the simplification in Fig. 10(b) (resp. Fig. 10(c), Fig. 10(d)) is feasible.

Now let's check the correctness of our construction. Let $p$ be the number of pentagons and $q$ the number of clauses. Then we could use a (hypothetical) algorithm for BUILDINGSIMPLIFICATION to answer the question: does there exist an $\varepsilon$-feasible solution with at most $K = 4p + 10q$ edges? If so, it means that all pentagons and clause gadget 14-gons can be simplified optimally. In this case all variable gadgets are in true or false state and, for each clause, there is at least one variable that does not trigger pressure, i.e., the corresponding literal is true. Thus the states of the variable gadgets yield a fulfilling truth assignment for the PLANAR3SAT instance $(U, C)$. On the other hand, if $(U, C)$ has a fulfilling truth assignment, we can set the variable gadgets to the corresponding states and are sure that in each clause gadget at least one of the three connectors does not transmit pressure, which means that all polygons can be simplified; exactly $K$ edges are needed in total. To summarize, the given instance of PLANAR3SAT is a yes-instance if and only if the instance of BUILDINGSIMPLIFICATION that we construct is a yes-instance, too.

Note that the connector is quite flexible. For example, we can make bends as shown in Fig. 9. Moreover, we can scale the pentagons parallel to their longest edge. Thus, it is possible to compensate for different edge lengths in the above-mentioned plane drawing of the variable-clause graph $G_{\mathrm{vc}}$. Now it is not hard to see that our reduction can be performed in polynomial time. $\square$



(a) state *true* (b) state *false*

Figure 8: The variable gadget. Shaded polygons define the input polygons. Black outlines show optimal simplifications.

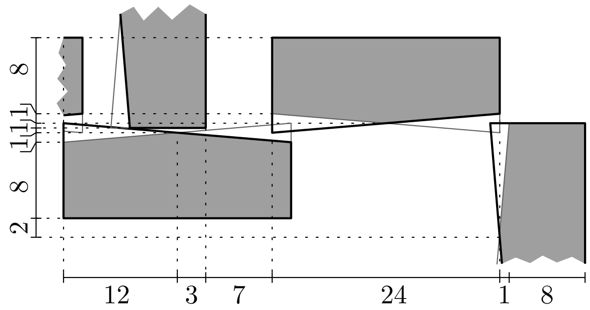



Figure 9: A detail of the variable gadget and a connector (see Fig. 11) with dimensions.

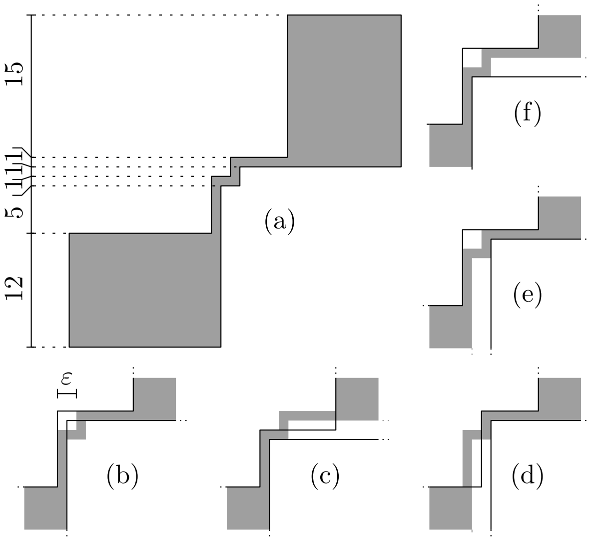
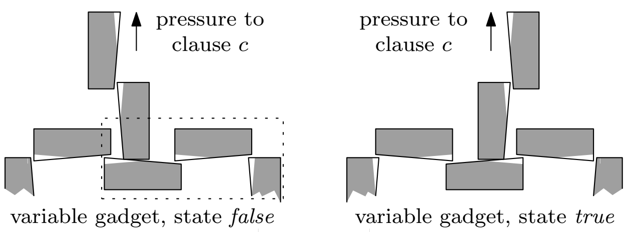



Figure 10: The clause gadget (a) has 14 vertices, each of its five optimal simplifications ((b)–(f)) has ten vertices.




(a) setting for $u \in c$.

(b) setting for $\bar{u} \in c$.

Figure 11: A connector that is attached to variable $u$. The region in the dotted rectangle is shown in Fig. 9 in detail.

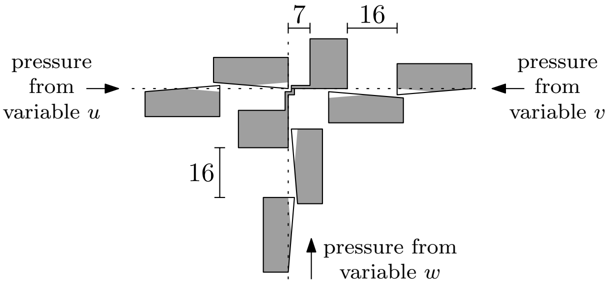
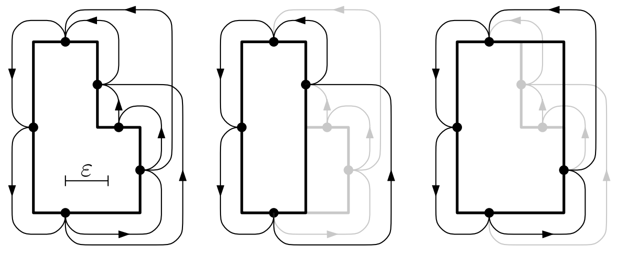



Figure 12: Three connectors attached to clause $\{u, v, w\}$. The dotted lines define the alignment of the connectors.




(a) input polygon and shortcut graph $G_{\mathrm{scut}}$

(b) solution and corresponding cycle in $G_{\mathrm{scut}}$

(c) another solution and corresponding cycle

Figure 13: A building with two feasible simplifications.

## 4 NEGLECTING THE SIMPLICITY REQUIREMENT

We now present an approach to solve the building simplification problem without insisting that the output polygons be simple. This relaxed problem can be solved efficiently. For this we search for a shortest cycle in a certain directed graph. We will use this approach in Sections 5 and 6 to also attack the original problem.

Given a building $P$ and a distance threshold $\varepsilon > 0$, let $E(P)$ be the set of edges of $P$ and let $S$ be the set of shortcuts of $P$ that satisfy the bandwidth criterion (requirement (R3)) and the intersection requirement (R2a). A shortcut $(e_k, e_l)$ satisfies (R2a) if and only if the L-shape of $(e_k, e_l)$ intersects $e_k$ and $e_l$. For example, the shortcut $(e_i, e_j)$ in Fig. 4(a) does *not* satisfy (R2a). Now we can define the (directed) *shortcut graph* $G_{\mathrm{scut}} = (E(P), S)$ with respect to $P$ and $\varepsilon$. For an example, see Fig. 13(a). Note that $S$ contains a shortcut for each pair of consecutive edges in $P$.

Testing whether a shortcut satisfies the bandwidth criterion can be done in $O(n)$ time, thus the shortcut graph $G_{\mathrm{scut}}$ can be constructed in $O(n \cdot |S|)$ time, where $|S| \in O(n^2)$ in the worst case. Figures 13(b) and 13(c) show simplified buildings; the selected elements of the shortcut graph, that is, edges and shortcuts, are highlighted. Observe that each feasible simplification corresponds to a cycle in $G_{\mathrm{scut}}$. Now an obvious idea is to search for a shortest cycle in $G_{\mathrm{scut}}$. Finding a shortest cycle in a digraph $G = (V, E)$ takes $O(|V| \cdot |E|)$ time (Itai and Rodeh, 1978). Note, however, that not all cycles of $G_{\mathrm{scut}}$ imply feasible solutions: a cycle can imply self-intersections and turning edges.

As the simplicity requirement (R1) renders the problem NP-hard, it is unlikely that there is an efficient algorithm that copes with dependencies between "distant" shortcuts (see Fig. 2). In contrast, the direction requirement (R2b) (see Fig. 4(b)) is concerned with pairs of *consecutive* shortcuts $(e_i, e_j)$ and $(e_j, e_k)$. Therefore we can take requirement (R2b) into account by extending our graph-based approach. To this end we introduce the graph $G_{\mathrm{succ}} = (S, A)$ whose arc set $A$ contains an arc for each pair of consecutive shortcuts $(e_i, e_j)$ and $(e_j, e_k)$ in $S$ if the sequence $(e_i, e_j, e_k)$ does not imply a change of the direction of $e_j$. Note that each arc $(s, t) \in A$ corresponds to one potential edge of the simplified building. A shortest cycle in $G_{\mathrm{succ}}$ yields a shortest cycle in $G_{\mathrm{scut}}$ that satisfies requirement (R2b). Applying the algorithm of Itai and Rodeh (1978) to $G_{\mathrm{succ}}$ takes $O(|S| \cdot |A|)$ time, where $|A| \in O(|S|^2)$ in the worst case. In practice, however, it is likely that the number of shortcuts and pairs of consecutive shortcuts is relatively small if the parameter $\varepsilon$ is set reasonably. Then one may expect that $|S|$ (or even $|A|$) is linear in $n$.

A simple way to speed up the procedure is as follows. The algorithm of Itai and Rodeh (1978) uses a subroutine DICIRCUIT. Given a graph $G = (V, E)$ and a node $v \in V$ it finds the shortest cycle through $v$ in $O(|V| + |E|)$ time; DICIRCUIT is applied to each node in $V$ to find the overall shortest cycle. For our problem, however, it is not necessary to apply DICIRCUIT to all nodes (that is, shortcuts) in $G_{\text{succ}}$. Let $e_j \in E(P)$ be an arbitrary edge of the polygon. In any case, the simplified polygon must contain a shortcut $(e_i, e_k)$ with $i \leq j < k$, that is, a shortcut starting at $e_j$ or omitting $e_j$. Thus, we need to apply DICIRCUIT only to a hopefully small set of shortcuts.

## 5 AN IP FOR BUILDING SIMPLIFICATION

The efficient algorithm from Sect. 4 can be used for a first attempt to solve the problem. If the result is feasible we have the globally optimal solution. In some cases, however, we can end up with a non-simple polygon. For such cases we suggest to apply mathematical programming techniques. The formulation that we present in this section is an *integer linear program* or simply *integer program* (IP). Basically, an IP is a linear program (LP) whose variables are constrained to integer values. While LPs can be solved in polynomial time, solving an IP is generally NP-hard. We need the integer variables to cope with the combinatorial nature of the NP-hard building simplification problem. For basics of mathematical programming and common solution techniques we refer to Papadimitriou and Steiglitz (1998). We have applied mathematical programming before to solve the aggregation problem in map generalization (Haunert and Wolff, 2006).

To express a solution of BUILDINGSIMPLIFICATION we use the definition of $G_{\text{scut}} = (E(P), S)$ according to Sect. 4. We introduce binary variables

$$x_s \in \{0, 1\} \quad \text{for each } s \in S$$

with $x_s = 1$ if and only if shortcut $s \in S$ is selected for the simplified building. Note that the number of selected shortcuts equals the number of edges in the simplified building, which we want to minimize. Thus we can express our objective as follows.

$$\text{Minimize} \quad \sum_{s \in S} x_s \,. \tag{1}$$

Next, we enforce that the union of the shortcuts $s$ with $x_s = 1$ define a cycle in the graph $G_{\text{scut}}$. We do this by ensuring that for each edge $e_j \in E(P)$ there is exactly one shortcut omitting $e_j$ or starting at $e_j$.

$$\sum_{s \in \left\{(e_i, e_k) \in S \,\middle|\, i \leq j < k\right\}} x_s = 1 \quad \text{for all } e_j \in E(P) \tag{2}$$

Next we enforce the simplicity requirement (R1). If we select a shortcut $(e_i, e_j)$, then part of its L-shape will belong to the simplified polygon. We can say this with certainty about the part between $e_i$ and $e_j$ (not including $e_i$ and $e_j$). If these parts intersect for a pair of shortcuts $s, t \in S$, we call the pair *conflicting* (see Fig. 2). We forbid that such pairs are selected.

$$x_s + x_t \leq 1 \quad \text{for all conflicting shortcuts } s, t \in S \tag{3}$$

Similarly, a shortcut $s \in S$ can imply an intersection with an original polygon edge $e \in E(P)$ (see Fig. 3). This case cannot be handled with constraint (3): generally, there is no shortcut $t \in S$ that would allow to conclude that a certain part of $e$ belongs to the simplified outline. In order to exclude this type of intersection, let $S_{s,e} \subseteq S$ be the set of shortcuts that omit $e$ or sufficiently shorten $e$. Now we introduce the following constraint.

$$\sum_{t \in S_{s,e}} x_t \geq x_s \quad \begin{array}{l}\text{for all pairs } (s, e) \in S \times E(P) \\ \text{where } s \text{ is in conflict with } e.\end{array} \tag{4}$$

This allows to select shortcut $s$ only together with a shortcut from the set $S_{s,e}$. For example, $(e_i, e_j)$ in Fig. 3 can be selected with $(e_k, e_l)$, which shortens $e_k$ and so avoids an intersection.

Finally, the direction requirement (R2b) can be subsumed by constraint (3), that is, we also call two consecutive shortcuts $(e_i, e_j)$ and $(e_j, e_k)$ conflicting, if the sequence $(e_i, e_j, e_k)$ implies that $e_j$ changes its direction.

Our IP has $|S|$ variables and $O(|S|^2)$ constraints (where $|S| \in O(n^2)$ in the worst case).

## 6 A HEURISTIC APPROACH

In order to obtain a polynomial-time performance, we propose a heuristic method. In other words, we give up the claim for exact optimality. Our approach is to iteratively solve the problem without considering the simplicity requirement. For this we apply the method from Sect. 4. Whenever we obtain a solution with intersecting polygon edges we remove an arc from the graph $G_{\text{succ}}$ and solve the problem again. Algorithm 1 defines our approach.

**Algorithm 1** Heuristic solution, iteratively removing arcs

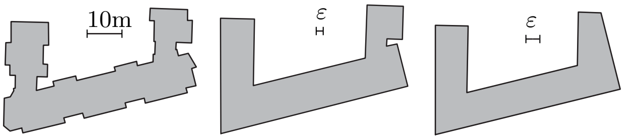
```
1: find a shortest cycle C in G_succ
2: while polygon P' corresponding to C is not simple do
3:     for each pair of intersecting edges e'_i and e'_j in P' do
4:         remove one arc from A that corresponds to e'_i or e'_j.
5:     end for
6:     find a shortest cycle C in G_succ
7: end while
```

Obviously the crucial decision is in line 4. A bad decision about which arc to remove from the graph can imply a suboptimal result. If the removal accidentally destroys the last directed cycle in the graph we even end up with an infeasible problem. To avoid this situation, we never remove an arc $(s, t)$ from $A$ if shortcuts $s$ and $t$ contain consecutive edges of the original building, that is, if $s = (e_{i-1}, e_i)$ and $t = (e_i, e_{i+1})$. As selecting the sequence $(e_{i-1}, e_i, e_{i+1})$ implies $e_i = e'_i$, this approach always allows to fall back to the original building outline. In the future we plan to define a non-unit cost for arcs that better reflects the objective of map generalization. Then it will be reasonable to remove the more expensive arc.

Finally, we sketch an exact variant of Algorithm 1. Instead of ultimately rejecting arc $(s_1, t_1)$ or $(s_2, t_2)$ when finding a conflict between them, we could divide the problem into two subproblems: one without arc $(s_1, t_1)$ and one without arc $(s_2, t_2)$. If we continue to branch like this we are sure to find the globally optimal solution. Though this exact procedure requires exponential time in the worst case, there is an interesting fact: under the condition that the number $c$ of conflicting edges is constant (or even polylogarithmic in $n$), we have an exact polynomial-time algorithm for BUILDINGSIMPLIFICATION. Such an algorithm is commonly referred to as *fixed-parameter algorithm*.

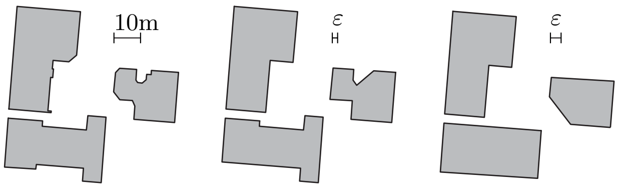
## 7 RESULTS

We implemented the methods from Sect. 4 and 5 as well as the heuristic from Sect. 6 in a Java application. Our program writes the IP formulation to a file and starts the free optimizer lp_solve (version 5.5.0.7) in a batch job; it reads the optimal solution from a file to construct the simplified shape. We also tested the commercial solver ILOG CPLEX® 9.1 with our IP formulation. Figure 14 shows a building with 68 vertices that we simplified with




(a) input (b) $\varepsilon = 2$ m (c) $\varepsilon = 4$ m

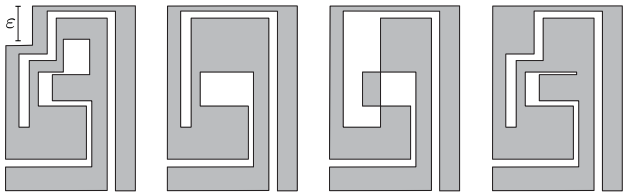
Figure 14: A building with two feasible simplifications.




(a) input (b) $\varepsilon = 2$ m (c) $\varepsilon = 4$ m

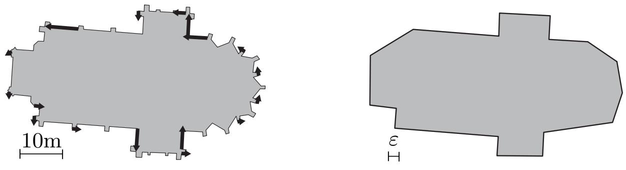
Figure 15: A set of buildings with two feasible simplifications.

this approach. Using CPLEX on a Linux server with 4 GB RAM and a 2.2 GHz AMD-CPU, the processing took less than 0.01s. The free optimizer lp_solve required 0.22s for the instance with $\varepsilon = 2$m and 0.17s with $\varepsilon = 4$m. The average processing time for the houses in Fig. 15 was 0.16s. Our heuristic performed similarly. The building in Fig. 14 was simplified in 0.06s with $\varepsilon = 2$m and 0.27s with $\varepsilon = 4$m. For all "real" buildings that we processed, the first shortest cycle that we found in the graph was feasible, so we always obtained the exact optimum. We were, however, able to construct a hypothetical building such that the heuristic produced a suboptimal result, see Fig. 16.




(a) input (b) IP solution (c) without simplicity constraint (d) heuristic solution

Figure 16: A hypothetical building with three simplifications.




(a) input (b) result with $\varepsilon = 2.5$ m

Figure 17: A simplified church. Edges that were selected for the simplified building are drawn as black arrows.

Our future research will deal with the definition of an appropriate optimization criterion. Uniform costs for edges are reasonable if the aim is data compression. This, however, is not the primary concern in map generalization. Figure 17 shows that the method does not distinguish important and unimportant edges: very short edges belonging to the pillars of the church were selected for the simplified shape. Since generalization aims to preserve important structures it is reasonable to charge a relatively high cost for selecting short edges. Generally, our heuristic allows to define any cost function for the arcs in $A$, that is, we can define additive costs for edges, shortcuts or pairs of consecutive shortcuts.

## 8 CONCLUSION

Our building simplification method produces ground plans that satisfy a given error tolerance while ensuring simplicity of polygons. As a basic optimization criterion the method minimizes the number of edges in the result. We have proved that the problem is NP-hard and therefore focused on integer programming and a heuristic approach. We conclude that both methods can be applied to solve problem instances that usually appear in practice. Our IP and our heuristic method can be used to simplify typical buildings in less that 0.1s; we consider this efficient enough for cartographic production. Only for an untypical, artificial shape our heuristic failed to find the optimal solution. Future research will show whether it is possible to incorporate a cost function into our model that better reflects the aims of map generalization, that is, to preserve important shape characteristics.

## References

Bard, S., 2004. Quality assessment of cartographic generalization. Transactions in GIS 8(1), pp. 63–81.

Bose, P., Cabello, S., Cheong, O., Gudmundsson, J., van Krefeld, M. and Speckmann, B., 2006. Area-preserving approximations of polygonal paths. Journal of Discrete Algorithms 4, pp. 554–556.

Campbell, G. M. and Cromley, R. G., 1991. Optimal simplification of cartographic lines using shortest-path formulations. The Journal of the Operational Research Society 42(9), pp. 793–802.

Chen, D., Daescu, O., Hershberger, J., Kogge, P., Mi, N. and Snoeyink, J., 2005. Polygonal path simplification with angle constraints. Computational Geometry: Theory and Applications 32(3), pp. 173–187.

de Berg, M., van Kreveld, M. and Schirra, S., 1998. Topologically correct subdivision simplification using the bandwidth criterion. Cartography and Geographic Information Systems 25(4), pp. 243–257.

Deveau, T., 1985. Reducing the number of points in a plane curve representation. In: Proceedings of Auto-Carto VII, Washington D.C., USA, pp. 152–160.

Gudmundsson, J., Narasimhan, G. and Smid, M., 2007. Distance-preserving approximations of polygonal paths. Computational Geometry: Theory and Applications 36(3), pp. 183–196.

Haunert, J.-H. and Wolff, A., 2006. Generalization of land cover maps by mixed integer programming. In: Proceedings of the 14th Annual International ACM Symposium on Advances in Geographic Information Systems (ACMGIS'06), Arlington, Virginia, USA, pp. 75–82.

Itai, A. and Rodeh, M., 1978. Finding a minimum circuit in a graph. SIAM Journal on Computing 7(4), pp. 413–423.

Kada, M. and Luo, F., 2006. Generalisation of building ground plans using half-spaces. In: Proceedings of the International Symposium on Geospatial Databases for Sustainable Development, Goa, India, ISPRS Technical Commission IV.

Knuth, D. E. and Raghunathan, A., 1992. The problem of compatible representatives. SIAM Journal on Discrete Mathematics 5(3), pp. 422–427.

Lichtenstein, D., 1982. Planar formulae and their uses. SIAM Journal on Computing 11(2), pp. 329–343.

Mayer, H., 1998. Model-generalization of building outlines based on scale-spaces and scale-space events. In: International Archives of Photogrammetry and Remote Sensing, Vol. 32 (3/1), pp. 530–536.

Papadimitriou, C. H. and Steiglitz, K., 1998. Combinatorial Optimization. Dover Publications, Mineola, NY.

Sester, M., 2005. Optimization approaches for generalization and data abstraction. International Journal of Geographical Information Science 19(8–9), pp. 871–897.

Staufenbiel, W., 1973. Zur Automation der Generalisierung topographischer Karten mit besonderer Berücksichtigung großmaßstäbiger Gebäudedarstellungen. PhD thesis, Technische Universität Hannover, Germany.